

VECTOR

APL Berlin 2000 Reports ...

- | | |
|----------------------------------|-----|
| • Cube Puzzle Solutions | 35 |
| • Objects in Dyalog-9 (Donnelly) | 74 |
| • RIP OOF (Kromberg) | 91 |
| • Reiter on CD Labels | 98 |
| • McDonnell on Apter's Puzzle | 116 |
| • De Kerf on Tribonacci | 131 |



*The Journal of the
British APL Association*

A Specialist Group of the British Computer Society

ISSN 0955-1433

www.vector.org.uk

Vol.17 No.2 October 2000

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 2000–2001

Category	Fee	Vectors	Passes
UK Private	£12	1	1
Overseas Private	£14	1	1
(Supplement for Airmail, not needed for Europe)	£4		
UK Corporate Membership	£100	10	5
Overseas Corporate	£135	10	
Sustaining	£430	10	5
Non-voting Member (Student, OAP, unemployed)	£6	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

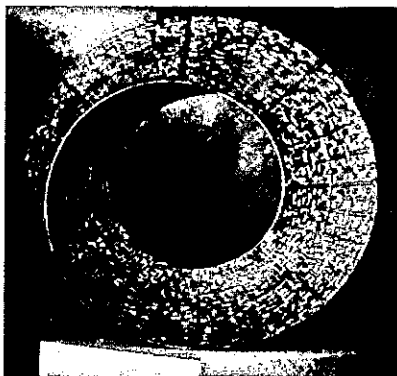
Advertisements in VECTOR should be submitted in typeset camera-ready format (A4 or A5) with a 20mm blank border after reduction. Illustrations should be photographs (b/w or colour prints) or line drawings. Rates (excl VAT) are £250 per full page, £125 for half-page or less (there is a £75 surcharge per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

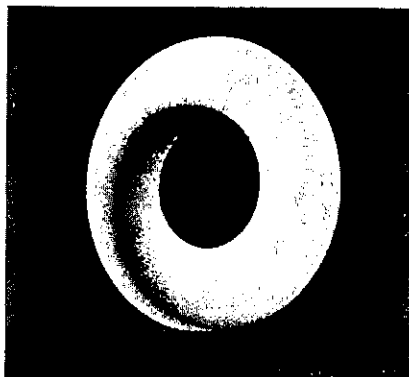
Email: apl385@compuserve.com.

Contents

		Page
Editorial	Stefano Lanzavecchia	3
APL NEWS		
Quick Reference Diary		5
British APL Association News		
BCS Specialist Groups Congress	Anthony Camacho	7
News from Sustaining Members		9
APL Product Guide	Gill Smith	10
The Education Vector		
Zark Newsletter Extracts	edited by Jon Sandles	25
Crossword		32
The Funny Cube: some solutions	Stefano Lanzavecchia	35
J-oltings 26: Here we go round ... and round and round ...	Norman Thomson	51
RECENT MEETINGS - APL Berlin 2000		
APL Berlin 2000 Reports	Stefano Lanzavecchia	56
Dyalog 9 Workshop Notes	Peter Donnelly	74
R.I.P. OOF	Morten Kromberg	91
GENERAL ARTICLES		
CD Labels and More	Cliff Reiter	98
TECHNICAL SECTION		
Hacker's Corner: How big is that Doggie in the Window?	Bill Parke, Adrian Smith	108
Technical Correspondence		112
At Play with J: Someone Just Moved! Who Was It? or, Apter's Puzzle	Gene McDonnell	116
The Generalized Tribonacci Sequence	Joseph De Kerf	131
Index to Advertisers		143



Umbilic Torus NC by
Helaman Ferguson, 1986



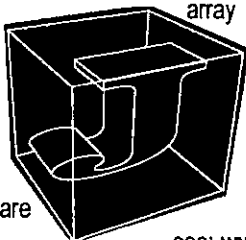
Umbilic Torus created in Jsoftware

For those who see the elegance in an algorithm, Jsoftware offers an OLAP programming language ideal for solving complex problems.

*Jsoftware is the modeling tool of choice
in the art of programming.*

Jsoftware is an OLAP language. Its high performance interactive nature allows a programmer to concisely express algorithms when analyzing complex data sets.

The core language is based on a small set of simple yet consistent rules with many powerful facilities to define new operations. Built-in functions are



highly optimized for operating on whole datasets at a time, allowing you to manipulate data far more easily than with conventional software. Also, sparse array support provides an efficient form of storage for very large data sets.

Make sure to join the JForum for a lively discussion on a variety of interesting J topics,

see: www.jsoftware.com/forum.htm

Strand Software, Inc.
19235 Covington Ct.
Shorewood, MN 55331
(612) 470-7345
Fax (612) 470-9202
info@jsoftware.com

www.jsoftware.com

Editorial

by Stefano Lanzavecchia (stf@apl.it)

*"Tears are falling, tears of joy"
Depeche Mode - from Black Celebration*

*"Ne aishitara daremo ga
Konna kodoku ni naru no?"
from Escaplowne - Yakusoku Wa Iranai*

Tears being shed, tears of joy. It feels like an ending. The euphoria in the comfortable air of the German night is not as contagious as it should be, according to the tacit rules of an official conference banquet. Somebody is trying to make a fool of himself with reasonable success. With the eyes closed, not only the buzz of the party, the explosions of laughter, the boast of a loud comment, the giggling, the hysteria of amusement reached the ears, but also the songs of the crickets in the park. An old castle, whose walls have silently watched hundreds, maybe thousands of parties. Or is it the voice of people trying to cancel their sorrows in the excitation of a glass of wine the real voice of the austere solid walls? On the morning after, the buzz remain in the ears, more like a bothering mosquito, while the fairy crickets sleep their rest. But now a bagpipe vibrates its minor harmonies like a cry, beating at the pulse of a large drum. The dancer moves and plays around the drone, followed by many eyes that dare graze her skin. Yet, another demoiselle's eyes, blue and cold, are nonchalantly piercing the night. Shivers down the spine. One instant of universal silence. But the night is still young. The proud princess will fail to meet the fearful knight, who hides in the long shadows cast by the low lights, while history repeats itself.

You are going to read more and perhaps more sound reports than mine about the APL 2000 conference that took place in Berlin this summer, that's why I'll keep it short. But let me add a few comments. First of all, the vendors: all of them showed once again how active they are, and in particular Soliton and Dyadic. I would say that the highlight of the conference was when Pete Donnelly of Dyadic presented his slide with the title "Bill's Telephone Call". APL has made it in the list of the languages to support and be supported by Microsoft's futuristic .NET (pronunciation: "dot net") platform and Pete's team will be responsible for this to happen. I will not try to hide my excitement about the platform itself and Dyadic's commitment but since it's a bit premature to talk about something that will not be available to the big public for at least another 18 months, I will not

spend more time on this. Rest assured, though, that we will keep you informed about the development of .NET. As a sidenote, I am glad to inform you that in the editorial for the September issue of the Microsoft Developer's Network magazine, APL was mentioned as being part of the background of the editor.

A few months ago I dismissed Sharp's APL for Linux as a product arrived too late and with too little to be truly interesting. I was wrong (and am now sorry to have been wrong). Soliton's presentations at the conference concentrated on their new interfaces with Java and Java's GUI codenamed Swing and proved that Sharp APL is a viable alternative on non Microsoft platforms to other languages: soon it will be equipped with a slick and functional platform independent (as platform independent as Java can be) IDE and a powerful graphical interface to build modern applications. A little disappointing was the participation, or the lack thereof of J Software's representative who seems to be more concentrated on their own conference (expect a report in a future issue of Vector) in Toronto, to be held in October. New array-oriented languages were introduced at the conference, like the interesting F-Script, derived from the object-oriented language Smalltalk, and this gives the impression that developers are beginning to realise that array paradigms can help solving a large class of problems. Personally I regret that the people behind Arthur Whitney's K do not spread their verb at APL conferences: I am fascinated by the language, and, while the user interface is so rough to be almost frightening, the power and the expressiveness of the language is surprising.

To stimulate feedback from the K community I decided to improvise as a K developer, and I propose a solution in K for Mr. Legrand's puzzle, as far as I know, the only existing solution in K.

APL DEVELOPER - South Africa

Full time salaried position in Johannesburg, South Africa, for APL developer.

You will be doing development, maintenance and support for a suite of financial modelling systems, developed in APL+Win (from APL2000) and with interfaces to VB.

Experience in APLGUI essential. Experience in the modelling of financial instruments highly advantageous.

Remuneration (in ZAR) based on qualifications and experience.

For further information, e-mail illsen@riskflow.com

Quick Reference Diary

Date	Venue	Event
13-15 Nov	Orlando	APL2000 User's Conference

The APL Berlin 2000 CD - Acknowledgements

Vector would like to thank the committee of APL Berlin 2000 for permission to distribute the working papers presented at the conference. The APL fonts on the CD remain the property of the copyright owners and are included solely for the purpose of allowing you to read the MS Word documents which are available as an alternative to the PDF versions of the documents. In particular, we would like to thank IBM Corp for extending their ACM permission to include the British APL Association in the list of organisations allowed to include copies of the APL2 fonts as part of published materials.

You may freely redistribute the PDF documents, but please mention APL Club Germany and the British APL Association on any copies you post on the Internet.

Dates for Future Issues of VECTOR

	Vol.17 No.3	Vol.17 No.4	Vol.18 No.1
Copy date	8th December	9th March	8th June
Ad booking	15th December	16th March	15th June
Ad Copy	22nd December	23rd March	22nd June
Distribution	January 2001	April 2001	July 2001

dyalog

APL

The Definitive APL for Windows™

The screenshot displays the Dyalog APL for Windows IDE interface. At the top is the title bar for the main window: "U:\HELP9\GUIREF (Manual.H.[Documents].[_Document].[Range])...". Below the title bar is a menu bar with "File", "Edit", "View", "Windows", "Session", "Log", "Action", "Options", "Tools", and "Help". A toolbar follows, containing icons for file operations, editing, and navigation. The main editor window shows the following code:

```
Dyalog APL/W Version 9.0.0
Serial No : 000042 / Pentium
Tue Jun 20 14:13:54 2000
clear ws
      )LOAD U:\HELP9\GUIREF
U:\HELP9\GUIREF saved Tue Jun 20
      )OBS
Help      Helpfiles      Manual  R
      )CS Manual
#.Manual
      )FNS
ANALYSE_CHANGES DISPLAY GET_APPLI
      UPDATE
Reading EVENTMAP from U:\HELP9\EU
```

The right side of the editor shows a list of line numbers and corresponding code snippets:

```
[0] R←GETXREF;W;REL;F;T;I;N
[1] OML←3
[2] DPATH←'†'
[3] R←0ρc'' 'COP,c''
[4] GET_EVENT_MAP
[5] 'H'DMC'OLECLIENT' 'Horc
[6] REL←W.Documents.Open'C
[7] W.Visible←1
[8] :With REL.Content
[9]     F←Find
[10]     F.Style←wdStyleHead
[11]     :While F.Found^F.Exe
[12]     T←Text
[13]     NAME TYPE←2↑(~T∈DAU[4 5
[14]     :If TYPE='Object'
```

Below the editor is the "Debugger - Manual.UPDATE [Tid:0]" window. It shows a list of code lines with line [9] selected:

```
[6] W.Visible←1
[7] :With REL.Content
[8]     F←Find
• [9]     F.Style←wdStyleHeading5
[10]     :While F.Found^F.Exe
[11]     T←Text
[12]     NAME TYPE←2↑(~T∈DAU[4 5
[13]     :If TYPE='Object'
```

The "Stack [Tid:0]" window shows the current stack frame: "UPDATE[9]*F.Style←wdS". At the bottom, the status bar displays "Cur-Obj: #.Manual.H (App) &:1", "DDQ:0", "DTRAP", "OSI:1", "OIO:1", and "OML:3".

Version 9 - an even better IDE(A)

<http://www.dyadic.com>

Dyalog Systems Limited, Riverside View, Basing Road, Old Basing, Basingstoke,
Hants. RG24 7AL, United Kingdom.

Tel:+44 1256 811125 Fax:+44 1256 811130 Email: sales@dyadic.com

Microsoft is a registered trademark and Windows and the Windows Logo are trademarks of Microsoft Corporation

British APL Association News

Visit to BCS Specialist Groups Congress

reported by Anthony Camacho

BCS SG congress

Ian Ritchie looked at how far computing has come in 50 years and where he thought it might be in 20 years' time. Some people have been remarkably prescient - for example Vannevar Bush in his article *As we might think* in 1945 - and some have done things much before their time - for example Doug Engelbart in the 1960s who more or less produced a personal computer environment with mouse on a mainframe. All the text and accompanying pictures we read in a lifetime can be held in 60-300 gigabytes; everything we say in about 15 gigabytes. These are well within the range of what personal computers can do. A lifetime's seeing (at video quality) would take about a petabyte - but we can expect this on a PC by 2020 or 2025.

Hamish Carmichael introduced the Computer Conservation SG, set up 11 years ago and supported by the London Science Museum, the Museum of Science and Industry in Manchester and the BCS. Some of the projects are: a replica of Colossus, a refurbished Pegasus, a rebuilt Turing 'bomb', an Elliott 401 and now some software such as George III.

Frances Freeman presented *Shopcreator* - the winner of the BCS awards last year. It is a simple way to create a web site from which things are to be sold, with all the abilities you need: security for credit card transactions, easy updating (if not done frequently the site loses credibility), automatic frequent re-registration of all site keywords with the major search engines and reliability (no downtime and no delays arising from server overload).

John Aeberhard told us about the Disability SG without visual aids or any variation of tone or pace.

After lunch Jennifer Stapleton asked us to consider the relationship between the BCS and our SGs and to let her know what we would like it to be.

Colin Thompson told us about BCS developments. These were mostly the consequence of the report of a working party, chaired by Brigadier Alan Pollard, that recommended:

- A wider range of membership options.
- That membership for Engineers, Managers and Teachers each be treated separately as the needs and resources appropriate to the three groups were so different.
- That there is a progression path for each class of member
- That the length of experience required (for the 'experience only' routes to membership is reduced to the minimum.
- That a wider range of academic and vocational qualifications are accepted in the examination route to membership.
- That people could join as a certified affiliate.

There were some frank admissions of weaknesses in the BCS and a new determination to define a structure that will overcome these and serve members, branches and SGs better.

Brian Layzell explained that the finance of the Technical Board (for SGs) was ring-fenced. He hoped the BCS would be able to handle Credit Card, Direct Debit and foreign cheque subscriptions for SGs and be able to hold records of their non-BCS members and do this maybe even if the SG year did not match the BCS year. The discretionary amount SGs can spend might be increased (for long it has been £500, which is unduly restrictive).

After Tea we broke into four groups to discuss (1) SGs and the Web, (2) Programme 2000 (mainly the Pollard report), (3) BCS/SG relations and (4) Finance.

I attended the third group: it was agreed that SGs have not always kept the BCS adequately informed of their activities and the BCS has not always kept SGs informed of BCS or other SG activities. If organised right SGs could use a BCS-provided admin back-up service, but the load may be very peaky. Some SGs would like the BCS to collect subscriptions and other payments from members as the variety of methods and currencies is hard to cater for. Some SGs (but not all) would welcome editorial or technical writing help.

Anthony Camacho
12 September 2000

Sustaining Members' News

Dyadic Systems Ltd

Dyadic is pleased to announce Dyalog APL/W Version 9 for Windows 95, Windows 98, Windows 2000 and Windows NT. This version replaces Dyalog APL/W Version 8.2.

New Features

Version 9 has a completely new Integrated Development Environment that employs dockable windows, an MDI Editor, and a host of new features. Docking also extends to Forms, SubForms, CoolBands and ToolControls so that you can provide docking facilities in your end-user application.

In Version 9, namespaces become truly first-class objects. This means that you can have an expression that evaluates to a namespace reference, and you can assign namespace references in the same way that you can assign variables and functions. Properties of GUI objects may now be assigned and referenced directly as if they were variables, and Methods (and events) may be invoked as if they were functions. This new syntax for manipulating GUI objects is faster and easier to use.

Version 9 includes much improved COM support which takes full advantage of namespace references. There are 5 new objects; Animation, BrowseBox, ComboEx, DateTimePicker and SysTrayItem and the capabilities of the Printer object have been improved.

Causeway

For the first time in recorded history, a new Dyalog release came and went without requiring a major rewrite from the Causeway company! As far as we can tell, CPro, RainPro and NewLeaf will run without modification in the new interpreter. Feedback from users would be welcome, and we will continue to ship our own applications on 8.2 for at least 6 months, but we will have "version-9 approved" releases on the website from 31st October. The ZIP passwords will change, so anyone who bought the tools in the last 12 months (or paid for an upgrade) should mail us for new passwords.

Look out for *HelpStuf* on the website anytime after VikAPL. We have been using this for our own manuals for years - now everyone can benefit!

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

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Dyadic	IBM RS/6000 MD320	11,736	APL POWERstation (Greyscale) 27.5 MIPS, 7.4 Mflops RISC Processor 8Mb RAM, 120Mb Disk 19" 1280x1024 Greyscale Graph Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD320	13,817	APL POWERstation (Colour) 27.5 MIPS, 7.4 Mflops RISC Processor 8Mb RAM, 120Mb Disk 16" 1280x1024 Colour Graphics Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD320	22,656	Advanced APL POWERstation 27.5 MIPS, 7.4 Mflops RISC Processor 16Mb RAM, 320Mb Disk, 150Mb Tape 16" 1280x1024 Colour Graphics Display AIX, OSF Motif, Dyalog APL (1-user)
	IBM RS/6000 MD520	37,114	APL POWERsystem (8-users) 27.5 MIPS, 7.4 Mflops RISC Processor 16Mb RAM, 320Mb Disk, 150Mb Tape CD-ROM Drive, 16 Ports AIX, Dyalog APL (2-8 user licence)
	IBM RS/6000 MD530	72,054	APL POWERsystem (16-users) 34.5 MIPS, 10.9 Mflops RISC Processor 32Mb RAM, 1.34Gb Disk, 2.3Gb Tape CD-ROM Drive, 16 Ports AIX, Dyalog APL (8+ user licence)
	IBM RS/6000 MD540	122,842	APL POWERsystem (32-users) 41 MIPS, 13 Mflops RISC Processor 64Mb RAM, 1.7Gb Disk, 2.3Gb Tape CD-ROM Drive, 32 Ports AIX, Dyalog APL (8+ user licence)
Optima	IBM Compatible	poa	Complete networked or stand-alone solutions including configuration installation, maintenance and commissioning.

APL INTERPRETERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Beautiful Systems	Dyalog APLW for Windows	poa	US Distributor of Dyalog APL products from Dyadic.
	Dyalog APL for Unix	poa	See Dyadic listing for product details.
The Bloomsbury Software Company	APL+PC Version 11	250	Upgrade to version 11 gives free runtime (£120 from any version)
	APL+Win v3.0	1350	A 32-bit Windows-hosted interpreter that runs under all Windows platforms including Windows 95. Note: Upgrade for £350 from version 1.8 or 2, £920 from version 1.0
	Migration to APL+Win	1060	from APL*PLUS/PC APL*PLUS/DOS any version. from earlier versions of APL*PLUS II
	APL+DOS	1250	APL*PLUS II DOS is renamed to APL+DOS.
	Migration to APL+DOS	760	from APL*PLUS/PC
	APL Link	200	Database Access
	APL Link Pro	500	
	APL*PLUS II for UNIX	poa	APL2000's 2nd generation APL for all major Sparc and Risc Unix workstations.
	APL*PLUS VMS	poa	2nd generation APL for DEC VAX computers under VMS.
	APL*PLUS Mainframe	poa	Enhances VS APL with many high performance, high productivity features. For VM/CMS and MVS/TSO offers simple upgrade from VS APL.
Dinosoft Oy	Dyalog APLW 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	
Dyadic	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.
	APL2 PC (US Version)	\$630	Product No. 5799-PGG. PRPQ Number RJ0411. Order from 1-800-IBM-CALL
	APL2 PC (European Version)	£348	Product No. 5604-290. Part number 38F1753. From all IBM dealers, including MicroAPL.
	APL2 for OS/2 Entry Edition	\$185	Part No 89G1556.
	APL2 for OS/2 Advanced Edition	\$650	Part No 89G1697. Contains all facilities of the Entry Edition plus: DB2 interface; co-operative processing TCP/IP interface; tools for writing APs; TIME facility
	APL2 for Windows version 1.0	\$1500	Product No. 5639-d46, part number 4229558.
	APL2 Runtime environment	\$250	Part No. 39L8419 - Packager and runtime CDs. One additional runtime install \$200, 5 additional \$900, 10 \$1,500.
	APL2 for Sun Solaris	\$1500	Product No. 5648-065.
	APL2 for AIX 6000	poa	Product No. 5765-012.
	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
Dyadic Systems Ltd.		poa	Leading distributor of Dyalog APL products in Denmark
IBM		poa	Leading distributor of IBM APL & GraphX products in Denmark
Inerson Software Inc.	J on the Web online registration ...		
	J Educational Edition	\$95	
	J Standard Edition	\$295	
	J Professional	\$895	
	Books and accessories (discounts for reg users)		
	J Dictionary	\$50	
	J User Manual	\$50	
J Phrases	\$50		

	J Primer	\$50	
	Concrete Math	\$40	
	Exploring Math	\$50	
	J User Conference Proceedings	\$35	
	Mugs, T-shirts, Mousepads	\$10 each	
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.
Lescasse Consulting	APL+PC	poa	Lescasse Consulting is the exclusive APL2000 distributor in France and also
	APL+Unix	poa	distributes in Switzerland and Belgium. Call for price quotes.
	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	APL.68000/X	1500-6000	Second-generation APL. Nested arrays, user defined operators, selective specification, etc. Multi-user AIX version with full OSF/Motif support.
	APL.68000 Level II Mac	520	Second generation APL. Full windowing interface. Hardware and software floating point support.
Oasis	Dyalog APL	poa	Dyadic Systems
	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	Fully fledged Windows development environment.
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.
	Dyadic and ISI products	poa	
	USA		
	Dyadic and ISI 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 Group (see Eventra)			
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.
The Bloomsbury Software Company (for VSAPL)	Enhancements & Sharefile	poa	Component files, quad-functions & nested arrays for VSAPL under VM/CMS & MVS/TSO
	Compiler	poa	The First APL compiler!
(for APL2)	Sharefile/AP	poa	STSC's shared access component file system for APL2. Comparable to all APL*PLUS file systems: multi-user storage of APL2 arrays with efficient disk usage.
Causeway	CausewayPro for Dyalog/W	400/\$600	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 <i>NewLeaf</i> 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 (1.8 and above)
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	HELM	poa	Decision Support System for top management. Handles large multi-dimensional tables, data analysis, EIS presentations, generates HTML and Latex output. Platforms: MS-DOS, LAN, Windows 3.1/95/NT. Ideal for APL customisation (APL*PLUS II and Dyalog APL); more than 150 installed.

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.
Eventra	DynaLink	poa	An ODBC client interface for DICE and IBM APL2 programs.
	Qualedi	\$1500-4000	Electronic Data Interchange (EDI) translation software for the PC, with strict compliance checking.
H.M.W.	4XTRA	poa	Front-end Foreign Exchange dealing / pos keeping
	Arbitrage	poa	Arbitrage modelling
	Basket	poa	Basket currency modelling
	Menu-Bar	poa	pull-down menu for APL*PLUS/PC
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 (AGSS)	\$250	for DOS, Product Number 5764-009
		\$500	for Workstations (OS/2, Aix, Solaris), Product Number 6764-092
		\$2500	for CMS, Product Number 5764-011
INFOSTROY	APL*PLUS/Xbase Interface (II/386 Version 2)	\$198	Complete package written in C. Comparable with the data, index & memo files of FoxPro, dBASE, & Clipper. Multi-user support. No DBMS license required.
	(DLL Version 1)	\$198	The same in a DLL form! Gives your Windows applications all advantages of DLLs.
Insight Systems	Causeway	poa	Leading distributor of Causeway products in Denmark <i>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!</i>
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	FRESCO Business Graphics	poa	Fast and easy business graphics DLL
	AP126/PC	poa	GDDM interface for Dyalog APL/W
	AP127/PC	poa	ODBC interface for Dyalog APL/W
	AP119/PC	poa	TCP/IP interface for Dyalog APL/W
	FACS	poa	EMMA-like interface to DB2 or ODBC databases
	ISAP	poa	MS Windows DLL for calling Dyalog APL/W from web pages via MS Internet Information Server. Visit www.lingo.com for a demo.
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.
Weighahead Systems	Weighahead 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
Adlee	Consultancy	poa	Development, maintenance, conversion, migration, documentation, of APL products in all APL environments
Ajay Askoolum	Consultancy	poa	APL+Win development and migration of actuarial, financial, mathematical applications.
Andrews	Consultancy	poa	APL programming and analysis, Year-2000 legacy systems, algorithms, tree-processing.
APL Solutions Inc	Consultancy	poa	APL systems design, development, maintenance, documentation, testing and training. Providing APL solutions since 1969.
AUSCAN Software	Consultancy	poa	APL software development, training
Bloomsbury Software	Consultancy	300-750+VAT	

Camacho	Consultancy	poa	Manuals; feasibility reports and estimates; analysis and programming; APL and MS Windows applications; Sharp, ISI APL, APL*PLUS, APL2/PC and other APLs spoken. Fixed price systems a speciality
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.
Dincoft Oy	Consultancy	poa	Specialised in very large databases.
Dittrich & Partner	Consultancy	poa	APL programming and analysis; APL workshops and training on the job
Dyadic	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.
H.M.W.	Consultancy	poa	System design consultancy, programming. HMW specialize in banking and prototyping work.
Hoekstra Systems Ltd	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	Moving applications between platforms. Client/server development. Multilingual user interface.
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-dimensional reports. Rapid custom software development based on proven modular toolset approach.
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 APLW. 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.
George MacLeod	Consultancy	poa	Design and programming of new APL applications. Enhancing and maintaining existing APL applications. Porting existing APL applications from one APL system to another. Supporting users of APL applications. Experienced on both mainframe, UNIX and PC APL interpreters.
Mackay Kinloch Ltd	Consultancy	from £40/hr	Design, analysis and programming for banking, insurance and pensions, financial planning and modelling, corporate performance and legal reporting
MicroAPL	Consultancy	poa	Technical & applications consultancy.
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)
Object Oriented Ltd	Consultancy	poa	General APL consulting, code recycling – mainframe to PC, performance tuning.
Omega Computing	Consultancy	poa	APL consultancy, programming, etc.
Optima	Consultancy	poa	A range of consultants specialising in all areas of pharmaceutical, industrial and financial systems with 5-15 yrs experience on both PC and mainframe.
RadSys Technologies	Consultancy	poa	Areas of expertise: financial systems, risk analysis systems, healthcare systems.
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, 20 years experience. Custom software development & training, PC and/or mainframe.
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 printers 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
Adlee	Employment	poa	Contractors and permanent employees
APL-385	Typefaces	poa	Variants of the APL2741 typeface available to specification.
Bloomsbury Software	Training	poa	Contact the company for details.
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 100612.1057@compuserve.com.
Oasis	Training	poa	Introductory courses in APL Advanced courses for different APL versions
Renaissance Data Systems	Booksellers		The widest range of APL books available anywhere. See Vector advertisements.

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 Club Germany	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)
Capital PCUG	Washington, D.C.	Monitor	Monthly meetings, occasional classes	free
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	SEAPL Newsletter	Quarterly meetings	\$10
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 Qlty SI-Info		SF60 (SI) + SF20 (SAUG)
Toronto SIG	Toronto, Canada	Gimme Arrays!	Monthly Meetings, APL skills database, J SIG, Toronto Toolkit	\$25

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	Marienhoehe 86, 25451 Quickborn, Germany. Tel: +49 4106 60977 Fax: +49 4106 67869 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: adsys@ibm.net
Adaytum Limited	Douglas Rowley	Castlegate, Tower Hill, BRISTOL BS12 0JA. Tel: 0117 921 5555 Fax: 0117 922 7749. Email:sales@adaytum.co.uk
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: 100331.644@compuserve.com
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 Club Austria	Harald F. Nelson	c/o N-TECH, Slebenbrunnenfeldg. 4-6, A-1050 Wien, Austria. Tel: +43 1 5458063 Fax: +43 1 5458063-17
APL Club Germany	Dieter Lattermann	Rheinstraße 23, D-69190 Walldorf, Germany. Tel: +49 6227-63469 Compuserve: 100332,1461
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: elandau@cais.com
Ajay Askoolum	Ajay Askoolum	42 Hanworth Road, Redhill, Surrey RH1 5HT Tel: 01737-771643 Email: ajay@askoolum.freeserve.co.uk
Association Francophone pour la promotion d'APL	Ludmila Lemagnen	174 Boulevard de Charonne, F-75020 Paris, FRANCE Email: lemagnen@aol.com
AUSCAN Software Ltd	Richard Procter	PO Box 39, Mansfield, Ontario L0N 1M0 Canada Tel: +1-705-434-1239 Email: rjp@interlog.com
BACUS	Joseph De Kerf	Rooienberg 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
Bloomsbury Software	Peter Day	Bloomsbury House, 74-77 Great Russell St., London WC1B 3DA, UK. Tel: +44(0)20 7436 9481 mobile +44(0)836 254660 Fax: +44(0)20 7436 0524 mobile +44(0)385 850629 Email: pd@bloomsbury-software.co.uk
Camacho	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS, UK. Tel: 0117-973 0036. email: acam@tesco.net
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: causeway@compuserve.com
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	1, Hefler Mill Cottages, Mosterton, Beaminstor, Dorset DT8 3HG, England. Tel: +44 (0)7931 370304. Email: earthspot@aol.com
CODEWORK	Mauro Guazzo	Corso Cairoli 32, 10123 Torino, Italy. Tel: +39 11 885168 Fax: +39 11 812 2652 Email: codework@inrete.it
ComLog Software	Jeff Pedneau	18728 Bloomfield Road, Olney, MD 20832 USA Tel: +1 (301) 260-1435 Email: jeff@softmed.com
CPCUG	Lynne Sturtz	Capital PC User Group, 51 Monroe Street, Suite PE-2, Rockville, Maryland 20850-2421, USA. Tel: +1 (301) 762-9372 Fax: (301) 762-9375.

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: dnosoft@dinosoft.fi
Dittrich & Partner Consulting GmbH	Axel Holzmciller	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
Dyadic Systems Ltd.	Peter Donnelly	Riverside View, Basing Road, Old Basing, Basingstoke, Hants RG24 0AL, UK. Tel: 01256-811125 Fax: 01256-811130
DynArray Corporation	Dr James Brown	16360 Monterey Rd. Suite 260, Morgan Hill, CA 95037, USA Tel: +1 (408)-782-8648 Fax: +1 (408)-782-6627 Email:info@DynArray.com
Eventra	Stuart Sawabini	Merritt Crossing, 440 Wheelers Farms Road, Milford, CT 06460, USA. Tel: +1(203) 882-9988. Fax: +1 (203) 882-9946 Email: ssawabini@eventra.com; eshaw@eventra.com
Evestic AB	Olle Evero	Berteliusvagen 12A, S-146 38 Tuillinge, Sweden Tel&Fax: +46 778 4410 Email: olle.evero@mailbox.swipnet.se
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, Westerland Road, LINTON, Kent ME17 4BS Tel: 01622 749328 Fax: 01622 749365 E-mail: martin@gsoft.freeseve.co.uk
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
H.M.W.Trading Systems	Chris Hogan	Hamilton House, 1 Temple Avenue, Victoria Embankment, London EC4Y 0HA, UK. Tel: 0870-1010-469; Email:HMWV@4tra.com
Hoekstra Systems Ltd	Bob Hoekstra	Dominique, Salsbury Road, Woking, Surrey, GU22 7UR, UK Tel: 01483-771028 Email: bob.hoekstra@kdtamsin.demon.co.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: 100612.1057@compuserve.com
IBM APL Products	Nancy Wheeler	APL Products, IBM Santa Teresa, Dept REN/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@vml.ibm.com
INFOSTROY	Alexei Miroshnikov	3 S. Tulenin Lane, St. Petersburg 191186 Russia. Tel:+7 812 312-2673 Fax:+7 812 311-2184 Email:aim@infostroy.spb.su
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
Iverson Software Inc.	Eric Iverson	33 Major Street, Toronto, Ontario, Canada M5S 2K9. Tel: +1 (416) 925-6096; Fax: +1 (416) 488-7559 Email: info@jsoftware.com
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: KYYY00361@niftyserve.or.jp
J Austria	Joachim Hoffmann	Flat 3, 42 Queen Annes Road, Bootham, YORK YO30 7AF Tel: 01904-651544 Email: joh@jaustria.freeseve.co.uk
KJK-tieto Oy	Kimmo Kekäläinen	Merikasarminkatu 10 B 56,00160 Helsinki, Finland. Tel: +358 50 55 27 207; Email: Kimmo.Kekalainen@pp.htv.fi
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: +33.1.46.05.10.78 Fax: +33.1.46.04.60.23 Email: eric@lescasse.com
Lingo Allegro USA, Inc	Steven J Halasz	1105 Chicago Avenue, Suite 155, Oak Park, IL 60302, USA. Tel:+1 800 546 4621-1 Email: sjh@sjhalasz.com
Lucas Solutions	Jim Lucas	Stubbedamsvej 9C, 3.tv., 3000 Helsingør, Denmark Tel: +45 49 26 52 42. Email: jel@danobs.dk

Mackay Kinloch Ltd	Alastair Kinloch	519 Webster's Land, Edinburgh EH1 2RX, Scotland, UK Tel: +44 (0)131 228 5235 Email: akinloch@globalnet.co.uk
George MacLeod	George MacLeod	37 Newhouse Rd, Bovingdon, Herts, HP3 0EJ, UK. Tel: 01442-831385 Fax: 01442-831388 Email: george.macleod@simcorp.com
Mercia Software Ltd.	Gareth Brentnail	Holt Court North, Heneage Street West, Aston Science Park, Birmingham B7 4AX, 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
Millinta Inc.	Dan Baronet	4215 St-Andre, Montreal, CANADA H2J 2Z3. Tel: +1 (514) 529-1434 Email: danb@millinta.com
Ellis Morgan	Ellis Morgan	Myrtle Farm, Winchester Road, Stroud, Petersfield, Hants GU32 3PE, UK. Tel: 01730-263843 Email: Ellis@mtlfrm.demon.co.uk
NY Slg	David Siegel	PO Box 2697, New York, NY10163-2697, USA. Email: NYSIGAPL@ACM.ORG
Oasis b.v.	Theo Zwart, Louis Rijkse	Lekstraat 4, 3433 ZB Nieuwegein, Holland. Tel: +31 30 60 66 336 Fax: +31 30 60 65 844 Email: info@oasis.nl or rijkse@oasis.nl
Object Oriented Ltd	Walter G. Fil	Am Grendel 2, CH-6004 Luzern, Switzerland. Tel: 41 41 418 70 70 Fax: 41 41 418 70 77 Email: info@object-oriented.com
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	115 Brighton Road, Purley, Surrey CR8 4HE, UK. Tel: +44 (0)20 8763 2490 Fax: +44 (0)20 8763 2491 Email: mailbox@optima-systems.co.uk
RadSys Technologies AB	Randolph Schrab	Lovsångarv. 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 313, Newtown, CT 06470, USA. Tel: +1 (203) 270-9729 Email: rendata@aplbooks.cnchost.com or orders@aplbooks.cnchost.com
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
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	Caseifa Postale 14343, 00100-Roma Trullo, Italy Email: marsac@vnet.ibm.com
SE APL Users Group	John Manges	413 Comanche Trail, Lawrenceville, GA 30044, USA Tel: +1 (770) 972-3755 Email: seapl@oc@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	Laurie Howard	Soliton Associates Ltd, Groot Blankenberg 53, 1082 AC Amsterdam, Netherlands Tel: +31 20 646 4475 Fax: +31 20 644 1206 Email: sales@soliton.com
SovAPL Russian Chapter of SIGAPL	Alexander Skomorokhov	PO Box 5061, Obninsk-S, Kaluga Region 249020, Russia Tel: +7(08439)31463 Fax: +1 (530) 504 8194 Email: askom@obninsk.com
Strand Software Inc	Anne Faust	19235 Covington Court, Shorewood MN 55331 USA Tel: +1 (612) 470-7345 Email: sales@jsoftware.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 Ulfhjelm	Novator Consulting Group AB, Svärtdvägen 11C, S-182 33 Danderyd Sweden. Tel: +46 8 622 63 50 Fax: +46 8 622 63 51 CServer: 100341,404
Swiss APL User Group		Swiss APL User Group, CH-3001, Bern 1, Switzerland Email: st@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) 672-7806

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/
Adaytum Limited	www.adaytum.com/
AFAPL	www.ensmp.fr/~scherer/tanglet/ (Journal available on line)
APL2000	www.APL2000.com/
APL-385	www.demon.co.uk/apl385/
APL Journal, Germany	www.rhombos.de/rb/apljourn.htm
AUSCAN	www.interlog.com/~rjp/auscan/
Eke van Batenburg	www.bio.LeidenUniv.nl/~Batenburg/index.html
Bloomsbury	www.bloomsbury.co.uk/software
Capital PC User Group	http://cpcug.org/
Causeway	www.causeway.co.uk/
CODEWORK	www.inrete.it/cdwwk/eng/homee.html
COSY (Bob Armstrong)	www.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/
Dyadic Systems Ltd	www.dyadic.com/
DynArray	www.dynarray.com/
Eventra	www.eventra.com/
FinnAPL	www.pyr.fi/apl/
Godin London Inc	www.godin.com/
Hoekstra Systems	www.khamsin.demon.co.uk/about/hsl/noframe.html
IBM APL2	www.ibm.com/software/ad/apl
Infostroy	www.insight.dk/infostroy/
Insight Systems ApS	www.insight.dk/
Iverson Software Inc	www.jsoftware.com/
Japan APL Association	www.naska.co.jp/JAPLA/
KJK-tieto Oy	www.kjk-tieto.com
Lescasse Consulting	www.lescasse.com/
Lingo Allegro USA Inc	www.lingo.com/
Mackay Kinloch	www.users.globalnet.co.uk/~akinloch/akinloch.htm
MicroAPL Ltd	www.microapl.co.uk/

Milinta Inc	www.milinta.com
Oasis b.v.	www.oasis.nl/
Optima Systems Ltd	www.optima-systems.co.uk
Renaissance Data	www.aplbooks.cnchost.com/
RE Time Tracker Oy	www.rett.fi/
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.jssoftware.com/
Rex Swain	www.rexswain.com/
Toronto SIG (for Toolkit)	www.torontoapl.org/
Jim Weigang	www.chilton.com/~jimw/
Weighahead Systems	www.weighahead.com

Vector Back Numbers

Back numbers of Vector are available from:

**British APL Association,
c/o Gill Smith,
Brook House, Gilling East,
YORK YO62 4JJ**

Price in UK: £10 per complete volume (4 issues);
£12 (overseas); £16 (airmail) including postage.

Zark Newsletter Extracts

introduced by Jon Sandles

Here is another Zark extract. Happily I can report that we have seen some interest in some of the challenges set in this regular feature and we have received excellent entries from Carlo Alberto and Stephen Jaffe. I have not had a chance to investigate how these compare to the Zark entries, but they look pretty good to me. If anyone fancies doing a more formal analysis, then please send it in.

From Stephen Jaffe (ExxonMobil Research)

In use at Mobil since 1974, modified several times as we migrated from APLSV to APLSF on DEC then to VMSAPL on VAX to APL*PLUS and finally to Dyalog.

```

v Z←X CMIOTA Y;I;B;C;M;N;L
[1]  a 1 For char matrices (∅IO is an implicit argument)
[2]  M←(∑2+1 1,ρX)ρX ∘ N←(∑2+1 1,ρY)ρY  a Let arguments to be matrices
[3]  C←∑1+(ρM)ρN
[4]  N←(∅C,1+ρN)+N ∘ M←(∅C,1+ρM)+M  a ... with the same no of cols
[5]  M←M∑' '  a Add row for "not found" case
[6]  Z←AI+AN∑M  a Sort (and grades) the together
[7]  a Z←AI+∅AVAN∑M  a Use this if APL version requires a collating seq.
[8]  a Z←AI+∅AV∑N∑M  a ... or this if it doesn't allow char grade (VSAPL)
[9]  B←I>1+ρN  a Select the items ∈ M
[10] L←(1+ρM)--∅IO  a Result if not found
[11] Z←((B/I-1+ρN),L)[∅IO+(+B)[(1+ρN)+Z]]  a See each item of N after
                                                which item of M was gone
[12] Z←L[Z+L×∑/M[Z;]∑N  a If they are equal this is
                                                the result, if not give L
[13] a ∘ (c)APLIT A FUB spi 13/09/1996 18:12 10363 D95
v

```

Utility Corner: Matrix Searching

In last issue's LIMBERING UP column, we asked for algorithms to search through character matrices. In particular, given two character matrices *A* and *B*, having the same number of columns, write a function whose syntax is

```
R←A CMIOTA B
```

Which returns a vector of row indices that give the locations in *A* where the rows of *B* first occur. You can consider the *CMIOTA* (Character Matrix IOTA) function an extension of the primitive APL function dyadic \vee on vectors *A* and *B*:

```
R←A ∨ B
```

Both functions search through *A* for the values in *B*. The result is a vector of indices whose length is determined by the shape of *B* and whose values are limited by the shape of *A*.

Frankly, we were surprised by the variety of algorithms received. There was nothing new, really, but we hadn't stopped to count the different approaches used to solve this age-old problem. More surprising were the results of our timings. Each algorithm was a winner. We timed the algorithms in three different APL environments, APL*PLUS PC, APL*PLUS II, and mainframe APL2 (the three environments under which Zark APL Tutor currently operates, naturally), and for a wide variety of argument sizes.

Some were better for skinny arguments, others for fat arguments. Some preferred small left arguments and lengthy right arguments, and some preferred the opposite. Some were good in one implementation of APL and lousy in another. Very interesting.

The results strengthened our conviction that the implementers of APL should consider a logical extension of dyadic ι to handle character matrix arguments. Let THEM decide which algorithm to use for different argument sizes. Besides, most vendors already provide an "assembled" function (named *LOOKUP*, *ROWFIND*, *IOTA* or some such) that performs this task. These assembled functions were not included in our timings. We suspect they would blow away the other APL algorithms.

Our recommendation: Use an assembled function if available to you. If not, pick the best algorithm from the ones below, given the sizes of your arguments.

The first three function are specific to APL*PLUS. They use the system function $\square DR$ (data representation) and exploit the fact that a two-, four-, or eight-column character matrix can be interpreted as a numeric vector. Once you view the array as a numeric vector, dyadic ι can be applied directly.

```

v R+A CMIOTA1 B
[1]  a For APL*PLUS PC, two-column args
[2]  a only (two-byte integers).
[3]  R+(163  $\square DR,A$ ) $\iota$ 163  $\square DR,B$ 
v

```

```

v R+A CMIOTA2 B; $\square CT$ 
[1]  a For APL*PLUS PC/II, eight-column
[2]  a args only (eight byte numbers).
[3]   $\square CT+0$  o R+(645  $\square DR,A$ ) $\iota$ 645  $\square DR,B$ 
v

```

```

v R←A CMIOTA3 B
[1]  a For APL*PLUS II, four-column args
[2]  a only (four-byte integers).
[3]  R←(323 ⍵DR,A)⍳323 ⍵DR,B
v

```

Notice *CMIOTA1* works for APL*PLUS PC but not APL*PLUS II, and *CMIOTA3* does the opposite, because APL*PLUS PC supports two-byte integers, whereas APL*PLUS II supports four-byte integers. Both APL*PLUS interpreters support eight-byte floating point numbers (*CMIOTA2*), but $\square CT$ (comparison tolerance) must be set to 0 so all 64 bits of significance are considered when dyadic ι makes its comparisons.

(The system variable $\square CT$ is consulted by the APL interpreter only when floating point comparisons are made. The value of $\square CT$ determines how fussy [some say "fuzzy"] APL will be when deciding whether two very close values are equal. For example, the expression $2.999999999999999=3.000000000000001$ will return 1 normally, but will return 0 if $\square CT$ is set to 0, its fussiest [least fuzzy] setting.)

The next two *CMIOTA* functions take a brute-force approach. *CMIOTA4* loops on the rows of the right argument, while *CMIOTA5* loops on the rows of the left argument. The heart of each algorithm is the use of the inner product $\wedge.=$ between a vector and the rows of a matrix. The result is a Boolean vector with as many bits as the matrix has rows. The ι bits flag rows that match the vector.

```

v R←A CMIOTA4 B;I;L;N
[1]  a Loop on rows of right arg.
[2]  R←(N+1+ρB)ρ0
[3]  L←(NρLOOP),0
[4]  →L[I+⍵IO]
[5]  LOOP:R[I]+(A∧.=B[I;])⍳1
[6]  →L[I+I+1]
v

v R←A CMIOTA5 B;I;L;M;N
[1]  a Loop on rows of left arg;
[2]  a decrement to return 1st, not last.
[3]  R←(N+1+ρB)ρI+⍵IO+M+(ρA)[⍵IO]
[4]  L←0,MρLOOP
[5]  →L[I]
[6]  LOOP:I←I-1
[7]  R[(B∧.=A[I;])/⍳N]+I
[8]  →L[I]
v

```

The next two functions are similar to the functions *CMIOA1*, *CMIOA2*, and *CMIOA3*, in that they convert the character matrices into numeric vectors so they can then apply dyadic \cdot . However, the process for converting to numbers is less direct. First, each distinct character is mapped to a distinct number. For example, all As are mapped to 1s, Bs to 2s, Cs to 3s, and so on. Then the numbers from each row are packed into a single number. Here's an illustration of the process on one row of a four-column matrix:

'FROG' → 6 18 15 7 → 6181507

In this case, the numbers are packed by multiplying them by respective powers of 100 and adding:

$$(6 \times 1000000) + (18 \times 10000) + (15 \times 100) + (7 \times 1)$$

The functions *CMIOA6* and *CMIOA7* are generalized implementations of this concept. In *CMIOA6*, the characters are converted to number by determining their index into $\square AV$, each character is translated into a number between 0 and 255 (its origin 0 index into $\square AV$). The resulting numbers are packed by multiplying by powers of 256.

```

v R←A CMIOA6 B;□CT
[1]  A Pack chars into numbers using
[2]  A □AV (max 6-7 cols)
[3]  □CT←0
[4]  R←(256⊥(□AV⊥&A)-□IO)⊥256⊥(□AV⊥&B)-□IO
v

```

There is a limitation to this approach. Since APL maintains a maximum of 16 or 17 significant digits for any number (that's all you can hold in eight-byte floating point representation), you'll start getting unreliable results when the number of digits exceeds 16 or 17. Since $256 \circ 1E17$ is 7.06, you shouldn't rely on this method for matrices with more than seven columns. Even then, make sure you set $\square CT$ to 0 to utilise the full significance.

But, hey! If we use a smaller packing factor than 256, we can pack more columns into the 16 or 17 digits of significance. It's not unreasonable to expect the character matrix arguments to contain only 20 to 30 different characters, not 256. Since $30 \circ 1E17$ is 11.51, a packing factor of 30 will allow you to use this packing method on matrices with as many as 11 columns. In *CMIOA7*, the two arguments are initially analysed to determine the list of distinct characters they use. The arguments are then translated to origin 0 indices into the distinct list, and the length of the list is used as the packing factor.

```

∇ R←A CMIOTA7 B;C;N;∅CT
[1]  A Pack chars into nums using distinct
[2]  A chars (max 6-15 or so columns)
[3]  ∅CT+0
[4]  N←ρC+∅AV~(∅AV~A)~B A Chars used
[5]  A N←ρC+((∅AVεA)∇∅AVεB)/∅AV A Ditto
[6]  R←(N∩(C∩&A)-∅IO)∩N∩(C∩&B)-∅IO
∇

```

The next two functions offer the maximum in elegance. Like *CMIOTA4* and *CMIOTA5*, they use inner product to compare the rows of their arguments. Unlike *CMIOTA4* and *CMIOTA5*, they do so without looping. All rows are compared at once. The "mismatch" inner product ($\vee.\neq$) is used instead of "match" ($\wedge.=$) for convenience, and the "leading" scan ($\wedge\backslash$) and the "how many" reduction ($+/\$) are used on the resulting Boolean matrix to count the number of mismatched rows before the first matching row.

```

∇ R←A CMIOTA8 B
[1]  A Inner product; transpose right arg.
[2]  R←∅IO++∧\A∇∇.≠&B
∇

```

```

∇ R←A CMIOTA9 B
[1]  A Inner product; transpose left arg.
[2]  R←∅IO++/\B∇∇.≠&A
∇

```

The differences between these two functions are the choice of argument to be transposed, and the choice of axis across which to scan and reduce.

Though elegant in their concise expressiveness, these two algorithms suffer from two weaknesses. First, all the rows of the left argument are compared to all the rows of the right argument. The algorithms aren't smart enough to stop when they find the appropriate match. Second. The result of the inner product is a Boolean matrix with as many elements as the PRODUCT of the number of rows in the two arguments. The matrix can get pretty big, sometimes big enough to generate a *WS FULL* error message.

The next function, *CMIOTA10*, is the most obscure. It relies on grade-up (\uparrow) to perform the comparisons between the two arguments. By concatenating the arguments row-wise, grading the concatenated matrix, and analysing the resulting grade vector, the desired *CMIOTA* result is constructed. Since grade-up is generally quite efficient compared to other searching algorithms, *CMIOTA10* promises to give the other functions a run for their money. Its only real disadvantage is its greater number of steps. For small arguments, the greater

amount of interpretation is likely to eat up the savings from its more efficient algorithm.

```

      V R←A CMIOTA10 B;I;G;N;S
[1]  A Gread-up char mats, using ⍵AV.
[2]  G←⍵AV⊆S+A, [⍵IO]B
[3]  S←S[G;]
[4]  A Flag 1st of distinct rows by shift
[5]  A and compare:
[6]  S←v/S×-11eS
[7]  A Ensure 1st elt is 1 (in case all
[8]  A rows the same):
[9]  S[⍵IO]+1
[10] A Indices of 1st distinct rows:
[11] I←S/G
[12] A Replicate for each like row:
[13] S[⍵IO]+⍵IO
[14] I←R+I[+⍵S]
[15] A Unsort inds (to catenated order):
[16] R[G]+I
[17] A Keep those corresp to right arg
[18] A and set 'not found' inds to
[19] A 'one greater than length':
[20] N←(ρA)[⍵IO]
[21] R←(N+R)[N+⍵IO]
      V

```

The final two functions use nested array approaches. Therefore, they require implementations of APL that support nested arrays. *CMIOTA11* works in any version of APL*PLUS that supports nested arrays, the notable exception being APL*PLUS PC. *CMIOTA12* works in APL2. The approach of both functions is the same: convert the matrix arguments into nested vectors of character vectors. Each row become one item in the resulting nested array. Then, use dyadic ι on the two vector arguments. Since dyadic ι has been extended to compare items of nested arrays, you get the desired result directly.

```

      V R←A CMIOTA11 B
[1]  A Uses APL*PLUS nests
[2]  R←(←A)⍵B
      V

      V R←A CMIOTA12 B
[1]  A Uses AP2 nests
[2]  R←(←[1+⍵IO]A)⍵←[1+⍵IO]B
      V

```


The comparative timings are tabulated in detail on the Vector web page. The point of these timings is to illustrate the KIND of effects you can expect to see as you vary the size of the arguments and as you move from one implementation of APL to another. Do you own timings if important.

Limbering Up: Comma-Delimited Data

When working with data from "external" (non-APL) applications, you need the ability to accept the data in a form comprehensible to both parties (you and them). One such form is the "comma-delimited file." The values in a comma-delimited file are separated by commas. The records are typically separated by a newline character (carriage return).

Here's a sample:

```
26,154.25,Fred
722,14.95,Jane
17,0,Bob
```

Each of these records contain three values of "fields" of information.

Here's your task: Given a comma- and newline-delimited character vector (newline at the end of each record), return a character matrix with the values realigned so they may be readily extracted via column-indexing. Like so:

```
26 154.25Fred
72214.95 Jane
17 0      Bob
```

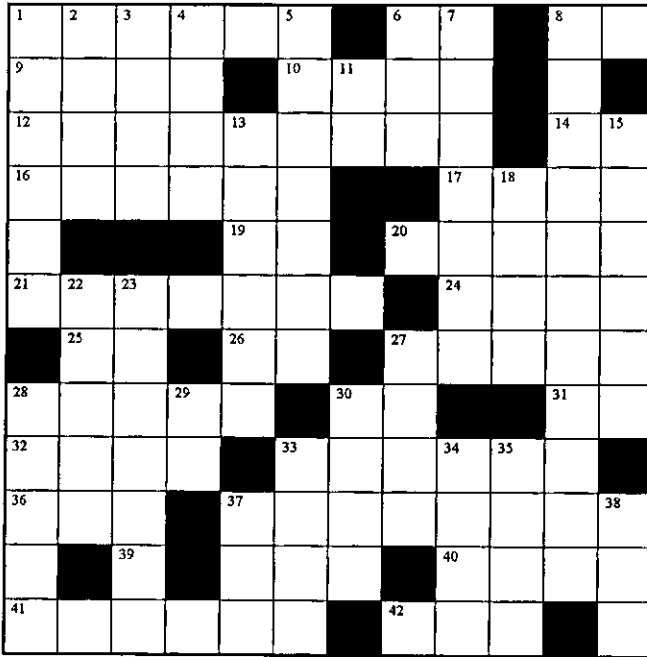
The function (call it *COMMAΔMAT*) should also return the global integer vector *fieldwidths* to tell you how wide the realigned fields are (3 6 4 in this case).

Write the fastest *COMMAΔMAT* you can muster and mail your solution on paper or APL*PLUS PC or APL*PLUS II disk to:

Vector Production
Brook House, Gilling East
YORK YO62 4JJ UK

Crossword

Crossword



Across

1. The first element of A , as a scalar
6. The values of x such that $(\wedge / (1 \rho F) \in x) \wedge ((\rho x) = \rho F) \wedge (1 + F[x]) \wedge . \geq ^{-1} + F[x]$
8. Resume the function that called the currently suspended function
9. The rank of the shape of F (or is it the shape of the rank?)
10. $(N > L) - N < L$
12. δ
14. $B \div |B + B = 0$
16. Clear the state indicator
17. $(-X \in DG) / X$
19. $1 \wedge B$
20. Subtract from the absolute value of M ...
21. Is (non-nested) A numeric?
24. $(12 + V), 12 + V$

25. The opposite of ' / '
26. M, all strung out
27. $\sqrt{}$, $Z1=20$
28. Add the magnitude of K to VM
30. The product of A and ...
31. The result of 9 across
32. $((\rho B)=+/B)\rho\Box IO$ for bit vector B
33. e to the power NVP, times five
36. $NX-1|NX$
37. Inverse of $+ \backslash V$ for vector V
39. The sin of one radian
40. $(NE+1)-1NE$ in origin 1
41. Assuming $V[\]+0$ and $\wedge/I\epsilon:\rho V$, do $V\leftarrow(1\rho V)\epsilon I$
42. Go to the line labelled L1

Down

1. $T \times 0$
2. $+ \backslash RR=RR$ for vector RR in origin 1
3. 10 if AE is numeric; ' ' if AE is character
4. $\Box NL \ 3$
5. Multiply alternate elements of A, whose shape is PTB, by the two values in M
6. ΨS without using Ψ
7. $10-FLEX \times |10 \div FLEX$
8. Branch to the line labelled P if D is positive, to Z if zero, to N if negative
11. IS the answer obvious?
13. $((SE \neq 0) > SE \in K) / SE$
15. $-V11-BG$
18. M2 is not an element of ...
22. $0=(M[;1] \wedge N[;1])-(M[;2] \wedge N[;2])$ for two-column matrices M and N
23. The sum of all but the first X elements of I, but no less than zero
27. $e+e$
28. ' ' $\rho \backslash V$
29. $(B \times B) \star .5$
30. $-A$
33. Pi less than 5
34. $\phi(-N)+L$
35. $((V0/1\rho V0), \Box IO+\rho V0)[\Box IO]$
37. V1 gets ...
38. MAT, less one dimension

Solution to Crossword in 17.1

^		ι	B	l	D		(1		A)
.	1	x	l	.	5	+	1	0	x	P	
=	/	N		5	2		-	/	B	L	T
	T		R	+		(A	D			?
(A	x	~	1	+	A	*	N)	÷	I
		ι	R	9		x	-	A	S	I	
	(I	+	1)	*	N		I		S
V	ι	2		÷		N)	(1	ρ
(1	3	x	4	8)	÷	5		5	2
)		ρ	5	x		A	↑	9		5
A	+	,	T		ι	K	-	V		+	/
?	2	5		(I	+	1)	*	-	N

The Funny Cube: Some Solutions

by Stefano Lanzavecchia and Adrian Smith

Introduction

The puzzle, proposed by Bernard Legrand and printed both on *Vector* Vol. 16 No. 3 and *Les Nouvelles D'APL* No. 32, is a simple yet interesting challenge. The attempts at cracking the problem of three French readers, including Mr. Legrand himself, have already been published by *Les Nouvelles D'APL* Vol. 33. It is our intention to review a few solutions from the readers of *Vector*.

Mr. Coquidé calculates the number of all the configurations admitted by the problem to be 122.880 in the simple case and 3.932.160 in the case where flipping of the pieces is allowed. Since these numbers are quite small for modern CPUs, it is possible to attempt a complete search of the configuration space. This strategy has in fact been followed by Mr. Legrand and by an unpublished attempt by Mr. Coquidé.

It is of course possible and easy, not to mention more efficient in terms of space and time used, to prune the search tree by removing early on impossible combinations (like 1 0 for Piece A and 3 0 for piece B in Mr. Legrand's notation), and this is the strategy used by almost all the other solvers with one noticeable exception.

Mike Day's solution [reviewed by S. Lanzavecchia]

The solver presented by Mr. Day is written in Dyalog APL, in a mixture of traditional APL2 code and Dyadic's Dynamic functions.

As the author points out, the code should be general enough to deal with more complex symmetries than the 3D cube, since the dependency on the geometry is only in the auxiliary functions *GetSides* and *AdjPlaceSides* and not in the solver itself. These claims were not actually verified by the author but the code looks nicely supportive.

The pieces are laid down one by one and before trying the next piece, all the impossible solutions are removed from the search tree to reduce the search space.

Because of this generality, which reflects itself in a clear, well structured code and compact code, if I had to choose a favourite solution, I would choose this one.

With its 440 milliseconds to solve the simple problem and 3400 milliseconds for the case with flipping allowed (on a PII 233 MHz running Dyalog APL 8.x), the solver shows that going for general code does not necessarily mean to compromise performance.

The listing

The code is written to run in `⎕IO+⎕ML+0`.

```
Solve+{⎕ Bin is 6 5 5 array of sides
  ⎕IO+0 ⋄ α+0      ⎕ default to no flips allowed
  canflip Bin+α ⎕
  npieces nsides+5 4
  sides+canflip GetSides Bin ⎕ 1 m f n array

  adjplacesides placeids+AdjPlaceSides 1
      ⎕ build place adjacency array
  A B C D E F+placeids
      ⎕ from here on the method is general for arbitrary npieces nsides

  (places placeflips placeturns)+{0 3}{1 4}{2 5}
      ⎕ names for column indexes
  placesandflips+places,[0.5]placeflips
      ⎕ column indexes in adjplacesides

  match-canflip MatchArray sides
      ⎕ build array of matching piece sides

  possiblestates+qsk1⌊*/sk+npieces,(canflip+1),nsides
      ⎕ really all possible state ROWS
  (pieces pieceflips pieceturns)+0 1 2
  piecesandflips+0 1      ⎕ column indexes in possiblestates

  r+adjplacesides SolveAll possiblestates
  r+(1,canflip,1)/+r
  r+r[;placeids;]
  r[&r[;;0];;]
}

r+adjplacesides SolveAll
possiblestates;statessofar;Check;remainingpieces;

newstates;newstate;stateid;state;nnew;adj;place;npieces
statessofar+1 1 3pA,0 0 ⋄ npieces+padjplacesides
Check+{
  adj+α
  state+⎕[adj[;places];]
  adj[;placeturns]+nsides|adj[;placeturns]+state[;;pieceturns]
  adj[;placesandflips]+state[;;piecesandflips]

  ^/match[+adj]
}
:For place :In 1+npieces      ⎕ [A] B D F C E seems best ordering
```

```

newstates+statessofar,[1]0 a newstates array needs one more set of rows
nnew+0
adj+adjplacesides[place]
:For stateid :In 1>statessofar

    remainingpieces+(1pieces)-statessofar[stateid;pieces]
    newstate+statessofar[stateid;],0
    :For state :In {w/1pw}possiblestates[;pieces]remainingpieces
        newstate[place;]+possiblestates[state;]
        :If adj Check newstate
            :If nnew=>r+pnewstates o newstates;+newstates
                a ++ expand newstates array if necessary
            :EndIf
            newstates[nnew;]+newstate o nnew+1
        :EndIf a Check
    :EndFor a state
:EndFor a stateid
statessofar+newstates[1nnew;]
:EndFor a place
r+statessofar

MatchArray+{a given l m f n array of sides
    a (ie l objects each with m sides
    a f flips (1 or 2) and n items
    a calculate (l m f)^2 Boolean array

    flip sides+a w
    length+3>0+sides
    corners+0,length-1
    insides+1+1+length
    mask+(11+pw)*.0+0>+w
    sidematch+{
        a[insides]^.=w[phiinsides]:0 a no match if lugs equal
        a[corners]^.=w[phicorners]:1 a then corners mustn't double up
        0 a else fail
    }
    sides*.sidematch sides
}

GetSides+{a+0
    flip Bin+a w
    length+1>pwBin
    up+Bin[;0;]
    down+phiBin[;length-1;]
    right+Bin[;length-1]
    left+phiBin[;0]
    r+up,[1]right,[1]down,[0.5]left
    *flip:r,[0.5]phiR[;phi+4;]
    rp=1|1 0 1 1\rp
}

AdjPlaceSides+{
    a list pairs of adjacent sides in layout
    a eg B3 adj A1

    a with layout 0

```

```

      A                      3 □ 1
      A                      2
      A NB backward looking (ie B to A but not A to B)
      A - can't check D:E if only done A B C
      .
      (A B D F C E)+i6      A was A B C D E F) + i6
      A this ordering seems better than i6 as it
      A eliminates more failures earlier in search

      Adj+6p0
      Adj[1]+c,c(B 3 A 1)
      Adj[2]+c(D 1 A 0)(D 0 B 0)
      Adj[3]+c(F 1 A 2)(F 2 B 2)
      Adj[4]+c(C 3 B 1)(C 0 D 3)(C 2 F 3)
      Adj[5]+c(E 1 A 3)(E 3 C 1)(E 0 D 2)(E 2 F 0)
      Adj+{(1 0 1 1 0 1\w)""Adj
      Adj(A B C D E F)
    )

```

Example of execution

Notice that the solution set is given in origin zero for the index of the pieces.

```

      Solve pieces
0 0
1 1
3 1
4 1
2 0
5 1

0 0
1 1
5 1
4 1
2 0
3 1
      p1 Solve pieces
52 6 3

```

Adrian's solution [reviewed by S. Lanzavecchia]

Adrian proposes a solution written in ISO APL. Unfortunately he only attempts to solve the simple case and leaves as an exercise for the reader the extension of his code to cope with the general problem.

The strong point of his solution is speed: less than 100 milliseconds on a PII 233 running Dyalog APL version 8.x.

The weak points, in my opinion, are the use of various global variables (which could easily become semi-globals) and its verbosity. Considering the very short

time it took him to come up with the solution and the fact that elegance on its own wasn't one of his goals, I must say that I am impressed.

If any of our readers think that they have a faster solution, please do not hesitate to send it for publication.

The listing

```

solutions+tiles bm
  A Start with the obvious stuff - match edges ignoring the corners
  A This makes an integer encoding of the edges to speed comparisons and save
  WS
  A The corners get used when we have 4 or more tiles down
  Δe+2 3 4 edges bm
  Δc+corners bm
  A Put them down in ABC... order to save thinking
  A We could do a little better with ABCDEF as this tests a corner sooner!
  cube1
  cube2
  cube3
  cube4
  cube5
  cube6
  A Return the result as required ...
  solutions+1 3 2φ((1+ρΔcn),2 6)ρΔcn,Δcr
  A Now how about an OpenGL representation (spinning of course) ...

r+cube1;ok
  A Place first tile ... 6 choices, 4 rotations
  A Array Δc holds the possibilities
  Δcc+24 4ρΔe[1;;(0 1 2 3φ4 4ρ14)]
  Δca+24 4ρΔe[2;;(0 1 2 3φ4 4ρ14)]
  Δce+24 4ρΔc[;(0 1 2 3φ4 4ρ14)]
  Δcn+,φ4 6ρ16
  Δcr+,6 4ρ0 1 2 3
  r+'After placing first tile, there could be '(v1+ρacc),' possibilities'
  A Arbitrarily choose the first one as canonical example ...
  ok+24+1
  Δcn+okfΔcn ◊ Δcr+okfΔcr ◊ Δcc+okfΔcc ◊ Δca+okfΔca ◊ Δce+okfΔce

r+cube2;cc;ca;cn;cr;ce;ok
  A Place Tile-B ... all options, then check valid combinations
  A and match on A2-B4 boundary
  cc+24 4ρΔe[1;;(0 1 2 3φ4 4ρ14)]
  ca+24 4ρΔe[2;;(0 1 2 3φ4 4ρ14)]
  ce+24 4ρΔc[;(0 1 2 3φ4 4ρ14)]
  cn+,φ4 6ρ16 ◊ Δcn+Δcn comb cn
  cr+,6 4ρ0 1 2 3 ◊ Δcr+Δcr comb cr
  ok+*/Δcn
  Δcc+Δcc mcomb cc
  Δca+Δca mcomb ca
  Δce+Δce mcomb ce
  Δcn+okfΔcn ◊ Δcr+okfΔcr ◊ Δcc+okfΔcc ◊ Δca+okfΔca ◊ Δce+okfΔce

```

```

r+cube3;cc;ca;cn;cr;ce;ok
  ok+acc[;2]=aca[;8]
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+'After placing 2nd tile, there are ',(v1+pacn),' possibilities'

```

```

r+cube3;cc;ca;cn;cr;ce;ok
  r+Place Tile-C ... all options, then check valid combinations
  r and match on E2-C4 boundary
  cc+24 4pae[1;;(0 1 2 3φ4 4p14)]
  ca+24 4pae[2;;(0 1 2 3φ4 4p14)]
  ce+24 4pac[(0 1 2 3φ4 4p14)]
  cn+,φ4 6p16 ◊ acn+acn mcomb cn
  cr+,6 4p0 1 2 3 ◊ acr+acr mcomb cr
  ok+acn[;1 2]*acn[;2p3]
  acc+acc mcomb cc
  aca+aca mcomb ca
  ace+ace mcomb ce
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+Now the boundary check ...
  ok+acc[;6]=aca[;12]
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+'After placing 3rd tile, there are ',(v1+pacn),' possibilities'

```

```

r+cube4;cc;ca;cn;cr;ce;ok
  r+Place Tile-D ... all options, then check valid combinations
  r and match on several boundaries
  cc+24 4pae[1;;(0 1 2 3φ4 4p14)]
  ca+24 4pae[2;;(0 1 2 3φ4 4p14)]
  ce+24 4pac[(0 1 2 3φ4 4p14)]
  cn+,φ4 6p16 ◊ acn+acn mcomb cn
  cr+,6 4p0 1 2 3 ◊ acr+acr mcomb cr
  ok+acn[;1 2 3]*acn[;3p4]
  acc+acc mcomb cc
  aca+aca mcomb ca
  ace+ace mcomb ce
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+Now the boundary check ...
  ok+acc[;1]=aca[;14]      r A1 == D2
  ok+ok^acc[;5]=aca[;13]  r B1 == D1
  ok+ok^acc[;9]=aca[;16]  r C1 == D4
  r+Finally we can check two corners ....
  ok+ok^1=+/ace[;2 5 14]
  ok+ok^1=+/ace[;6 9 13]
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+'After placing 4th tile, there are ',(v1+pacn),' possibilities'

```

```

r+cube4;cc;ca;cn;cr;ce;ok
  r+Place Tile-D ... all options, then check valid combinations
  r and match on several boundaries
  cc+24 4pae[1;;(0 1 2 3φ4 4p14)]
  ca+24 4pae[2;;(0 1 2 3φ4 4p14)]
  ce+24 4pac[(0 1 2 3φ4 4p14)]
  cn+,φ4 6p16 ◊ acn+acn mcomb cn
  cr+,6 4p0 1 2 3 ◊ acr+acr mcomb cr
  ok+acn[;1 2 3]*acn[;3p4]
  acc+acc mcomb cc
  aca+aca mcomb ca
  ace+ace mcomb ce
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+Now the boundary check ...
  ok+acc[;1]=aca[;14]      r A1 == D2
  ok+ok^acc[;5]=aca[;13]  r B1 == D1
  ok+ok^acc[;9]=aca[;16]  r C1 == D4
  r+Finally we can check two corners ....
  ok+ok^1=+/ace[;2 5 14]
  ok+ok^1=+/ace[;6 9 13]
  acn+ok/acn ◊ acr+ok/acr ◊ acc+ok/acc ◊ aca+ok/aca ◊ ace+ok/ace
  r+'After placing 4th tile, there are ',(v1+pacn),' possibilities'

```

```

r+cube5;cc;ca;cn;cr;ce;ok
  r+Place Tile-E ... all options, then check valid combinations
  r and match on several boundaries
  cc+24 4pae[1;;(0 1 2 3φ4 4p14)]
  ca+24 4pae[2;;(0 1 2 3φ4 4p14)]
  ce+24 4pac[(0 1 2 3φ4 4p14)]
  cn+,φ4 6p16 ◊ acn+acn mcomb cn
  cr+,6 4p0 1 2 3 ◊ acr+acr mcomb cr
  ok+acn[;1 2 3 4]*acn[;4p5]

```

```

acc+acc mcomb cc
aca+aca mcomb ca
ace+ace mcomb ce
acn-ok^acn o acr-ok^acr o acc-ok^acc o aca-ok^aca o ace-ok^ace
R Now the boundary checks ...
ok+acc[;4]=aca[;18]      A A4 == E2
ok+ok^acc[;17]=aca[;15]  A E1 == D3
ok+ok^acc[;10]=aca[;20]  A C2 == E4
R ... and remaining edge checks around the top
ok+ok^1=+/Ace[;10 17 16]
ok+ok^1=+/Ace[;18 1 15]
acn-ok^acn o acr-ok^acr o acc-ok^acc o aca-ok^aca o ace-ok^ace
r+'After placing 5th tile, there are ',(v1+pacn),' possibilities'

```

```

r+cube6;cc;ca;cn;cr;ce;ok
R Place Tile-F ... all options, then check valid combinations
R and match on several boundaries
cc+24 4pae[1;;(0 1 2 3φ4 4p14)]
ca+24 4pae[2;;(0 1 2 3φ4 4p14)]
ce+24 4pac[;(0 1 2 3φ4 4p14)]
cn+,φ4 6p16 o acn+acn mcomb cn
cr+,6 4p0 1 2 3 o acr+acr mcomb cr
ok+^/acn[;1 2 3 4 5]=acn[;5p6]
acc+acc mcomb cc
aca+aca mcomb ca
ace+ace mcomb ce
acn-ok^acn o acr-ok^acr o acc-ok^acc o aca-ok^aca o ace-ok^ace
R Now the boundary checks ...
ok+acc[;21]=aca[;19]      A F1 == E3
ok+ok^acc[;22]=aca[;3]   A F2 == A3
ok+ok^acc[;23]=aca[;7]   A F3 == B3
ok+ok^acc[;24]=aca[;11]  A F4 == C3
R All 4 corners must be happy this time!
ok+ok^1=+/Ace[;21 11 20]
ok+ok^1=+/Ace[;22 19 4]
ok+ok^1=+/Ace[;23 3 8]
ok+ok^1=+/Ace[;24 12 7]
acn-ok^acn o acr-ok^acr o acc-ok^acc o aca-ok^aca o ace-ok^ace
r+'After placing 6th tile, there are ',(v1+pacn),' possibilities'

```

```

em+inx edges cd
R Take cube data and code the 24 edges so we can match them quickly
R <inx> gives the notes to check - initially we just use the middle three
R We need both clockwise and anti-clockwise mappings
R Now we just need to match integers on each boundary!
em+2 6 4p0
R Work around top,left,bottom,right recording the +ve direction
em[1;;1]+21^acd[;1;inx]  A Top
em[1;;2]+21^acd[;inx;5]  A Right
em[1;;3]+21^acd[;5;φinx] A Bottom
em[1;;4]+21^acd[;φinx;1] A Left
R Now the same ordering of edges but encoded not-backwards ...
em[2;;1]+21-^acd[;1;φinx] A Top
em[2;;2]+21-^acd[;φinx;5] A Right
em[2;;3]+21-^acd[;5;inx]  A Bottom

```

```

em[2;:4]+21~bcd[;inx;1]  a Left

cm+corners cd
a Take cube data and grab the 8 corners so we can match them quickly
cm+6 4p0
a Work around top,left,bottom,right recording the +ve direction
cm[;1]+cd[;1;1]      a Top left
cm[;2]+cd[;1;5]      a Top Right
cm[;3]+cd[;5;5]      a Bottom Right
cm[;4]+cd[;5;1]      a Bottom Left

m+x comb y;s
s+(p,x)*p,y
m+((p,y)/x),[1.5]s py

m+x mcomb y;s
s+(1+p*x)*1+py
m+((1+py)*x),(s,1+py)py

```

Example of execution

```

tiles cubedata  a Solves the 'this way up' case of the cube puzzle
After placing first tile, there could be 24 possibilities
After placing 2nd tile, there are 11 possibilities
After placing 3rd tile, there are 90 possibilities
After placing 4th tile, there are 44 possibilities
After placing 5th tile, there are 12 possibilities
After placing 6th tile, there are 2 possibilities
1 0
2 1
4 1
5 1
3 0
6 1

1 0
2 1
6 1
5 1
3 0
4 1

```

Stefano's solution [presented by the author]

Since this is my very first attempt at writing a program in K, language that I attempted to learn in the very process of solving the puzzle, I will not claim that my K coding style is typical, efficient or that, in general, it gives a good example of a K program. In fact I would appreciate if some K expert could send in his solution to the puzzle or at least send comments about my code.

My first attempt at solving the puzzle had been done in APL, in purely functional style using Dyadic APL's dynamic functions and it wasn't very efficient, nor elegant. A few months later I decided to try another array oriented language and I

rewrote the original algorithm in K. Since the differences between K and APL are fundamental and I had lost the workspace anyway, this is not a translation of the APL code, but more a rethinking.

The strategy

Internally the pieces are represented as a four-element list containing the position of the ones of the top, right, bottom and left edge respectively. The local variable `rffaces` contains a representation of the pieces in all the possible states of rotation and flipping for fast access.

Instead of exploring the entire search tree I decided to cut impossible branches as soon as possible, by imposing edge and corner conditions. The two Boolean-valued functions `pfe` and `pfc` are used to identify and remove with the aid of the operator `filter` the invalid combinations. The function `newfs` prepares a new set of possible solutions by adding to all the partial solutions a yet unused piece in all its rotation and flip states.

The main execution loop is written in terms of a reduction of list of function applications to the initial state `0 0 0`, which in the internal zero-origin notation corresponds to piece A, non-flipped, not-rotated. A new piece is placed and then all the non-conformant solutions are removed by applying conditions at the edges and at the corners.

Please notice that the solver can be used also for given sets of pieces with a bigger or smaller side, since nowhere in the code there is an explicit reference to 5. When I wrote my solver I hadn't seen Mike Day's submission. The strategy is similar, but his code is more general. It would be a good challenge to try and adapt his ideas to my framework written in K.

No global variables or global utility functions are used.

Speed

To solve the general problem, with flipping of the pieces, my code takes a bit less than 1500 milliseconds on a PII 233 running K's interpreter version 2.8t (don't miss your free download from <http://www.kx.com/>). This is likely to be the fastest execution time among all the solvers collected by Vector and Les Nouvelles D'APL for the general problem. If Adrian's code were adapted to solve the general problem, it would most likely be faster by at least a factor of 5. My code seems to be the most compact of all, as well.

The listing

Since the solutions in the case where flipping is not allowed are a subset of the general case, I only show the code to solve the general case. It would be trivial to specialise to code to solve the simplified version of the problem.

```

/ Solve the Funny Cube Puzzle by Bernard Legrand

littlecube: {[pieces]
  abs: { (x*x>0) + (-x<0)*x }
  filter: {[f;y] y[& f'y]}

  newfs: {
    try: {
      tryflip: { ,/((!6) _dv/ x),\:/:~!2 }
              :,/ (tryflip x),\:/:~!4
    }
    :,{x(x,,y)/:try x[;0]}'x
  }

n: -1 + *1 _ ^pieces
fcs: { (&'y[0,x;],+y[;0,x])[0 3 1 2] }
faces: n fcs'pieces
rfaces: n {(y;abs (x,0,x,0)-y[0 3 2 1])}'faces
rffaces: n {3 (abs {0,x,0,x}-1!)\ y}'rfaces
pfe: {[y;e1;e2;f1] (#k)-#?k:(rffaces . y[e1[0]],e1[1]),abs f1 - rffaces
. y[e2[0]],e2[1] ] }
pfc: {[y;c1;c2;c3;m] 1=+/m _in'(rffaces . y[c1[0]],c1[1];rffaces .
y[c2[0]],c2[1];rffaces . y[c3[0]],c3[1])}
fitedge: {[fs;e1;e2;f1] filter[pfe{;e1;e2;f1};fs] }
fitcorn: {[fs;c1;c2;c3;m] filter[pfc{;c1;c2;c3;m};fs]}

y: (,,0 0 0) {y@x}/ ( newfs
  fitedge[;0 1;1 3;0]
  newfs
  fitedge[;1 1;2 3;0]
  newfs
  fitedge[;2 0;3 3;0]
  fitedge[;0 0;3 1;n]
  fitedge[;1 0;3 0;n]
  newfs
  fitedge[;0 3;4 1;0]
  fitedge[;3 2;4 0;0]
  fitedge[;2 1;4 3;0]
  fitcorn[;3 2;4 0;2 1;0 0 0]
  fitcorn[;4 0;3 1;0 0;n,n,0]
  fitcorn[;0 0;1 0;3 0;n,0,n]
  fitcorn[;1 0;3 0;2 0;n,0 0]
  newfs
  fitedge[;0 2;5 1;0]
  fitedge[;1 2;5 2;n]
  fitedge[;2 2;5 3;n]
  fitedge[;4 2;5 0;0]
  fitcorn[;4 2;5 1;0 2;n,0 0]
  fitcorn[;2 1;5 3;4 3;n,0,n]
  fitcorn[;1 1;2 2;5 2;n,0 0]
  fitcorn[;0 1;1 3;5 1;n,n,n])
:1 0 0+:/:~!y
}

```

Example of execution

Instead of showing all the solutions, which would take up a lot of space, we calculate their number.

```
sol: littlecube pieces
^sol
52 6 3
```

Let's check the first solution in the solution set:

```
*sol
(1 0 0
 2 0 1
 3 1 0
 4 0 2
 6 1 1
 5 1 0)
```

Finally, let's extract the subset of solutions that do not require flipping of the pieces and verify that they match Mr. Legrand's proposed solution:

```
sol[&0=+/'sol[;;1]]
((1 0 0
 2 0 1
 4 0 1
 5 0 1
 3 0 0
 6 0 1)
(1 0 0
 2 0 1
 6 0 1
 5 0 1
 3 0 0
 4 0 1))
```

A Solution from Jacques Grenot [notes by Adrian Smith]

This solution arrived as an ATF file, so I guess it came from an APL2 original. I no longer have easy access to APL2, but fortunately APL+Win comes with JIN as standard and was able to read and run the code with no problems. The timing was run on an AMD K6, running at 200MHz, so we are fairly comparable. The following are Jaques own notes on the strategy.

Strategy

Face A is fixed. From the remaining 5 faces (10 if flipping) find an E such that column 5 of E locks into column 1 of A. From the remaining faces, find a D and an F such that they match E above and below (they are then in the agreed conventional state). Find a C matching E to the left, and check whether the remaining face meets the requirements for B. Note the condition at corners, where 3 edges meet -one tooth and two notches!

Of course this is NOT the only way. Is it a reasonable way? Have I failed to see the obvious? That much, at least, is now out of my system. I hope you will have as much fun taking it apart as I had putting it together.

Jacques Grenot
 12 Tobruk Avenue
 Cremorne 2090, Australia Email: jgrenot@bigpond.net.au

The Code

Function LeCube is invoqued with a FLIP left argument (No flip if 0) and the array Abin, defining the 6 notched faces, in state 0, with which to build a cube.

```

v Z+Flip LeCube
Abin:FACE;STATE;S16;S4x5;S4x9;S4x13;S4x16;Slist;A;No_Flip;Ix;Bix;Cix;Lix;Six;Zix
;C_n;C_sol;Cbin;Vbin;EDGE1;EDGE2;EDGE3;EDGE4;Sel_E;Sel_D;Sel_F;Sel_C;Sel_B;Msk_E
;Msk_D;Msk_F;Msk_C;Msk_B;E_vec;D_vec;F_vec;C_vec;B_vec;B;BI;BJ;BIJ;C;CI;CJ;CIJ;D
;DI;DJ;DIJ;E;EI;EJ;EIJ;F;FI;FJ;FIJ
[1]  a Le petit cube de Monsieur Legrand (Vector V16 No.3)
[2]  a
[3]  a Fix utility functions
[4]  a
[5]  []IO+1
[6]  [FX]"1"+""2+(1+/'Δ'_g[OCR 'LeCube'])<<[[]IO+1][OCR 'LeCube']
[7]  A+1(:5)(:5)[Abin
[8]  No_Flip+Flip=0
[9]  Six+1+1(5+5*Flip)
[10] C_n+0
[11] Cbin+Abin
[12] Z+' '
[13] ~No_Flip/L00
[14] a
[15] Lix+7 8 9 10 11 2 3 4 5 6
[16] Zix+c[2]10 2ρ2 0 3 0 4 0 5 0 6 0 2 1 3 1 4 1 5 1 6 1
[17] Cbin+Cbin,[1](1+1:5)(:5)(:5)[φAbin
[18] a
[19] L00:
[20] Vbin+FACE"1(6+5*Flip)
[21] EDGE1+S4x5"Six
[22] EDGE2+S4x9"Six
[23] a
[24] Sel_E+cA[;1]
[25] Msk_E+(0vA[1;1]),1 1 1,0vA[5;1]      a mask possible zero corners
[26] E_vec+SlistA/"Msk_E/"Sel_E"=EDGE1
[27] ~(0<ρE_vec)/NEXT_EI
[28] a
[29] NEXT_EI:
[30] ~(0=ρE_vec)/DONE      a DONE - we have processed all possible E edges
[31] EIJ+1+E_vec          a take edge list for next face
[32] E_vec+1+E_vec       a Adjust E_list
[33] EI+1+EIJ           a Face index
[34] EIJ+1+EIJ         a List of edges on this face
[35] NEXT_EJ:
[36] ~(0=ρEIJ)/NEXT_EI  a We have exhausted this side
[37] EJ+4|1+1+EIJ       a *** Adjust to state
[38] EIJ+1+EIJ          a Update remaining edges list

```



```

[39] E+EI STATE EJ
[40] a *****
[41] a
[42] a For this E_A pair, find possible D and F matches
[43] a
[44] a Now form Sel_D and scan for potential D
[45] Sel_D+A[1;5 4 3 2],(A[1;1]vE[1;5]),E[1;4 3 2 1] a match for D
[46] Msk_D+(0vSel_D[1]),(7p1),(0vSel_D[9])
[47] D_vec+Slist^/"Msk_D/"(cSel_D)*">EDGE2
[48] -(0<pD_vec)/NEXT_DI
[49] a
[50] NEXT_DI:
[51] -(0=pD_vec)/NEXT_EJ
[52] DIJ++D_vec
[53] D_vec+1+D_vec
[54] DI++DIJ
[55] Ix+EI
[56] -No_Flip/L01
[57] Ix+Ix,Lix[Ix-1]
[58] L01:
[59] -(0<+/DI=Ix)/NEXT_DI a Face already selected
[60] DIJ+1+DIJ
[61] NEXT_DJ:
[62] -(0=pDIJ)/NEXT_DI
[63] a final D state is DJ+4|DJ-1
[64] DJ+4|_1+DIJ a *** Adjust to state
[65] DIJ+1+DIJ
[66] D+DI STATE DJ
[67] a *****
[68] a
[69] a Now form Sel_F and scan for potential F
[70] a
[71] Sel_F+E[5;1 2 3 4],(E[5;5]vA[5;1]),A[5;2 3 4 5] a match for D
[72] Msk_F+(0vSel_F[1]),(7p1),(0vSel_F[9])
[73] F_vec+Slist^/"Msk_F/"(cSel_F)*">EDGE2
[74] -(0<pF_vec)/NEXT_FI
[75] a
[76] NEXT_FI:
[77] -(0=pF_vec)/NEXT_DJ
[78] FIJ++F_vec
[79] F_vec+1+F_vec
[80] FI++FIJ
[81] Ix+EI DI
[82] -No_Flip/L02
[83] Ix+Ix,Lix[Ix-1]
[84] L02:
[85] -(0<+/FI=Ix)/NEXT_FI
[86] FIJ+1+FIJ
[87] NEXT_FJ:
[88] -(0=pFIJ)/NEXT_FI
[89] a Final F state is FJ
[90] FJ++FIJ a *** No adjustment required to state
[91] FIJ+1+FIJ
[92] F+FI STATE FJ
[93] a *****
[94] a
[95] Sel_C+D[1;4;1],(D[5;1]vE[1;1]),E[2 3 4;1],(E[5;1]vF[1;1]),F[2 3 4 5;1]
[96] Msk_C+(0vSel_C[1]),(11p1),(0vSel_C[13])
[97] Ix+DI EI FI
[98] -No_Flip/L03

```

```

[99] Ix+Ix,Lix[Ix-1]
[100] L03:
[101] Cix+Six-Ix
[102] EDGE3+S4x13"Cix
[103] C_vec+Slist^/"Msk_C/"(cSel_C)*">EDGE3
[104] -(0<pC_vec)/NEXT_CI
[105] a
[106] NEXT_CI:
[107] +(0=pC_vec)/NEXT_FJ
[108] CIJ+>C_vec
[109] C_vec+1+C_vec
[110] CI-Cix["1+>CIJ]
[111] CIJ+1+CIJ
[112] NEXT_CJ:
[113] +(0=pCIJ)/NEXT_CI
[114] CJ+>CIJ a *** No adjustment required to state
[115] CIJ+1+CIJ
[116] C->CI STATE CJ
[117] a *****
[118] a
[119] Sel_B+(D[1;5]vA[1;5]),D[1;4 3 2],(D[1;1]vC[1;1]),C[2 3 4;1]
[120] Sel_B+Sel_B,(C[5;1]vF[5;1]),F[5;2 3 4],(F[5;5]vA[5;5]),A[4 3 2;5]
[121] Ix->CI
[122] ->No_Flip/L04
[123] Ix+Ix,Lix[Ix-1]
[124] L04:
[125] EDGE4+S4x16"Bix+Cix-Ix
[126] B_vec+Slist^/"(cSel_B)*">EDGE4
[127] +(0=pB_vec)/NEXT_CJ
[128] BIJ+>B_vec
[129] B_vec+1+B_vec
[130] BI-Bix["1+>BIJ]
[131] BIJ+1+BIJ
[132] NEXT_BJ:
[133] +(0=pBIJ)/NEXT_CJ
[134] BJ+>BIJ
[135] BIJ+1+BIJ
[136] C_n->C_n+1
[137] ->No_Flip/L05
[138] C_sol+(2 3p' ' ' ' ' 1 0 0),[1](>Zix["1+BI CI DI EI FI]),BJ CJ DJ EJ FJ
[139] ->L06
[140] L05:
[141] C_sol+7 2p' ' ' ' ' 1 0 BI BJ CI CJ DI DJ EI EJ FI FJ
[142] L06:
[143] a []->C_n C_sol
[144] Z+Z,>C_n C_sol
[145] ->NEXT_FJ
[146] a
[147] DONE:
[148] 'All E matches exhausted...I think!...'
[149] Z->Z
[150] ->0
[151] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[152] A^_01:
[153] Z+S16 I
[154] U+I->Vbin
[155] Z+U[1;],U[1+14;5],(phiU[5;14]),U[4 3 2;1]
[156] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[157] A^_02:
[158] Z+S4x5 I;U

```

```

[159] a Returns four borders in clockwise sequence
[160] U+S16 I
[161] Z+S+''U(4φU)(8φU)(12φU)
[162] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[163] Δ^-03:
[164] Z+S4x9 I;U
[165] a Four string of two consecutive borders
[166] U+U,U+S16 I
[167] Z+(9+U)(9+4+U)(9+8+U)(9+12+U)
[168] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[169] Δ^-04:
[170] Z+S4x13 I;U
[171] a Four string of three consecutive borders
[172] U+U,U+S16 I
[173] Z+(13+U)(13+4+U)(13+8+U)(13+12+U)
[174] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[175] Δ^-05:
[176] Z+S4x16 I;U
[177] a Four string of all four borders shifted one at a time
[178] U+U,U+S16 I
[179] Z+(16+U)(16+4+U)(16+8+U)(16+12+U)
[180] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[181] Δ^-06:
[182] Z+FACE I
[183] Z+I(15)(15)[]Cbin a No_Flip 6 faces --- Flip 11 faces
[184] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[185] Δ^-07:
[186] Z+I STATE J;U;T
[187] U+FACE I
[188] T←0
[189] ROTATE:
[190] →(T=J)/END
[191] T←T+1
[192] U+φφU
[193] →ROTATE
[194] END:
[195] Z←U
[196] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[197] Δ^-08:
[198] Z←Slist M;K;L;ZL
[199] a from the 5x4 matrix of matching edges per side,
[200] a produce the vector list of (Side,Edge list)
[201] Z←''
[202] K←1
[203] L←0 1 2 3
[204] →(2=ρρM)/Enc_M
[205] M←1(←ρM)ρM
[206] Enc_M:
[207] M←c[2]M
[208] Next:
[209] →(0=ρM)/0
[210] K←K+1
[211] ZL←(←M)/L
[212] M←1+M
[213] →(0=ρZL)/Next
[214] Z←Z,cK,ZL
[215] →Next
[216] AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
V

```

Monsieur Legrand has provided a set of conventions for the representation of solutions. Two programs are suggested; a basic one, monadic, which does not allow the flipping of faces, and a more general program which uses a left argument to determine whether flipping is allowed. As the second program includes the first as a subset, I have dispensed with its transmission. If I have correctly interpreted the instructions, and if my logic is both correct and correctly programmed, there are two solutions to the No-flip case and 43 solutions if flipping is allowed. Note that FACE 4 flipped is equivalent to FACE 4 in STATE 2... The results are presented as by Mr Legrand's conventions, but preceded by a solution number.

```
tt←[wcall 'GetTickCount' ◦ qq*0 LeCube Abin ◦ ([wcall 'GetTickCount')]-tt
All E matches exhausted...I think!...
```

```
169      qq
1
  1 0
  2 1
  6 1
  5 1
  3 0
  4 1
2
  1 0
  2 1
  4 1
  5 1
  3 0
  6 1
```

Speed

As you can see, the function runs in around 170ms, and the corresponding solution for the 'Flip' case runs in 734ms, making it one of the faster examples we have. Just for interest, I took out the bizarre 'hide the subfunctions' approach (delocalise the subfunctions, run it, kill line-6 and the bottom of the listing) and the times came down to 106ms and 680ms respectively. Maybe APL2 has a much faster `⊞FX`, but even so it is hard to see the benefit of this approach.

J-ottings 26: Here we go round ... and round and round ...

by Norman Thomson

Do you feel mildly irritated when a report says something like "the figures may not add up to exactly 100 because of rounding"? I do. For one thing it would probably be just as easy to make the figures add up as to make the excuse, and in any case surely the whole essence of rounding is to *make* figures tally. You have probably guessed the next bit is going to be "it's easy in J", and yes, you are right - moreover it provides a nice illustration of the development of a simple, but not trivially simple, J verb. So let's begin by breaking down the process of rounding.

Consider the problem of rounding a value to a given number, say n , of decimal places. The simplest approach involves a three stage process. In the first stage the number is *raised* by moving the decimal point n positions to the right, the second stage consists of *swinging* the resulting value up or down to the nearest integer according to whether the fractional part of the resulting number is above or below 0.5, and the third stage reverses the first stage, that is it *lowers* the numbers by moving them n positions to the left. "Moving the decimal point" is in turn a process which, in more exact terms, consists of multiplying by 10 to the power n , with positive n meaning move to the right and negative n meaning move to the left. These decimal point movements can be summarised as

<code>pow=.10&^</code>	NB. 10 to the power
<code>raise=.*pow)~</code>	NB. 1 raise 2.57 is 25.7
<code>lower=.%pow)~</code>	NB. 1 Lower 25.7 is 2.57

Next let's sharpen the *swinging* process. Again this is a sequence of two simpler processes, the first consisting of adding 0.5, and the second taking the floor:

<code>swingu=.<.@+&0.5</code>	NB. move to nearest integer
<code>swingu 4.4 5.6 2.5</code>	
4 6 3	

Notice that 2.5 goes up to 3 - more of that later. At first sight the symmetry of the three stage process - raise, swing, lower - might suggest that these three verbs composed as a fork would be appropriate. However maturer consideration shows that this is not the case, since the essential operation of a fork can be summarised informally as: first execute the left and right prongs concurrently, then execute the middle prong on the transformed data, whereas in this case the lowering can only

take place *after* the other two processes have completed, and so raise and lower are *not* concurrent.

There *is* in fact a fork present in the rounding process, but it is the *lower* sub-process which forms its central prong, the other prongs consisting of (1) the raising and swinging sub-process in sequence, and (2) the re-extraction of *n*, the number of decimal places which was "consumed" in the raising sub-process.

This analysis leads to the following development of a rounding algorithm:

```
rnd=. [Lower swingu@raise NB. syntax is 1 rnd 2.57
```

The reason for the 'u' in swingu is that swinging possesses a degree of asymmetry in that a number with a fractional part which is exactly equal to 0.5 is rounded up. The "mirror image" verb

```
swingd=.<.@&0.5 NB. move to nearest integer, 0.5 goes down
swingd 4.4 5.6 2.5
4 6 2
```

has the opposite effect. For most practical purposes the two can be used interchangeably, that is *rnd* is equivalent to

```
rndd=. [Lower swingd@raise NB. syntax is 1 rndd 2.57
```

In the days before computers, the practice was sometimes followed of rounding exact halves to the nearest even integer in the hope that any errors so arising might roughly balance out. A "neutral" swing verb *rndn* incorporating this can be constructed using a gerund which is J's mechanism for the case statement.

```
isodd=.2&| NB. 1 for an odd integer, 0 for even
getfrac=.1&| NB. Get the fractional part of a number
swing=.swingd`swingu @.(isodd&(-getfrac))
```

If *swing* is used the `(isodd&(-getfrac))` decision has to be made separately for each value in a vector and so the *rndn* must be written

```
rndn=. [lower swing each@raise NB. syntax is 1 rndn 2.57
```

where `each=.&>`.

Now think about how to extend this basic rounding algorithm to do *balanced* rounding, that is the process of making the rounded values of a vector tally

exactly to their rounded total. This problem happens typically with percentages, for example the three values in

```
a=.36.24 29.73 34.03
+/a
100
```

total exactly 100, but the individual values round to 36.2 29.7 and 34.0 which total 99.9 .

```
1 rnd a
36.2 29.7 34
```

Begin by considering what adjustment might sensibly be made in the absence of a calculating device. This would probably be something like: following basic rounding, look for the value whose fractional part after raising comes closest to 0.5 (that is the 36.24) and round that one up. If the deficit between the rounded total and 100 had been 0.2 say, the two values nearest to 0.5 would be rounded up, and so on. Since this procedure happens strictly in the swinging stage, the structure of an algorithm for balanced rounding is going to differ only in one of the verbs already developed for rounding, so write it as

```
brnd=. [lower balance@raise NB. Balanced round
```

The problem is then reduced to that of sharpening what is meant by the verb "balance". Take *swingu* as a starting point. Instead of adding 0.5 indiscriminately to all the floors in preparation for lowering, it would be better if all raised values were initially truncated (that is floored), and then a value of 1 added to as many as are necessary to fill the gap between the sum of the raised values (1000) and the sum of their floors (999). Call this gap the rounding gap which, translating the previous sentence into J, is (+/) - +/@<. .

Clearly the order of candidature for adding a 1 is that of the size of the fractional part and so a verb must be constructed which determines the ranking of the fractional parts.

Define

```
getfrac=.1&|
rkd=: /:@\:
rkd 4 2 7 1
1 2 0 3
```

NB. get fractional part
NB. rank a vector downwards
NB. eg. 7=max and so has rank 0

and compose these two verbs, which is incidentally a nice little illustration of the use of the conjunction "with" (&) :

```
(rkd&frac)1 raise a
1 0 2
```

This says that, in the present example, the biggest fractional part is associated with the first term. Since the rounding gap is 1, this is the only item to which 1 will be added at the swing stage, and so the result of the comparison `rkd&getfrac < calcrgap` will supply exactly the right mix of 1s and 0s to add prior to lowering.

For a the rounded total is an integer and so too is the rounding gap. When this is not the case as with

```
b
1.631 0.478 1.939 4.236
2 raise b
163.1 47.8 193.9 423.6
+/2 raise b
828.4
+/.2 raise b
826
```

where the rounding gap is 2.4, only two of the items require 1 to be added, whereas if the rounding gap had been 2.6 then the round of the sum would be 8.29 and *three* of the items would require 1. Thus the rounding gap itself should be swung in order that the balance property be fulfilled that the sum of the rounded values should exactly equal the round of the sum of the original values. This leads to the definition of the verb

```
calcrgap=.swingu@(+/-) - +/@<. NB. Calculate rounding gap
```

and the verb `balance` completes the operation of adding 1 to qualifying items before lowering

```
balance=:(rkd&getfrac < calcrgap) + < . NB. adjusted floor
```

All the elements of `brnd` are now in place and so its definition is complete.

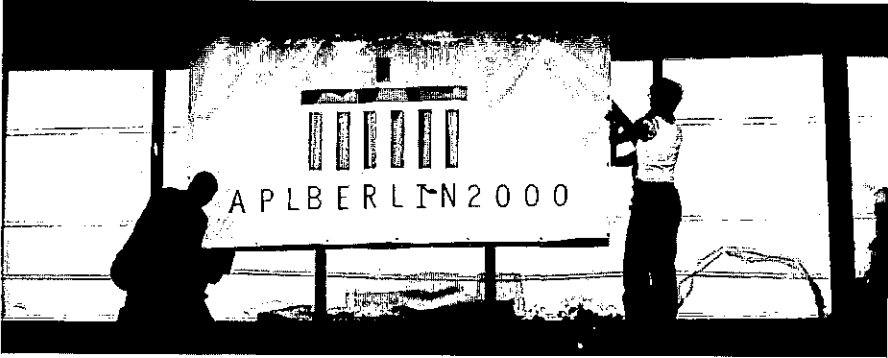
```
2 brnd b
1.63 0.48 1.94 4.23
(+/b),(+/2 rnd b),+/2 brnd b
8.284 8.29 8.28
```


Finally here is everything brought together to bring this well-rounded article (pun intended!) to a close:

```
rnd=[lower swingu@raise
      raise=(*pow)-
      pow=.10&^
      swingu=<.@+&0.5
      lower=(%pow)-
      NB. normal rounding
      NB. multiply y.by 10^x.
      NB. 10 to the power y.
      NB. adjust y.to nrst integer
      NB. divide y.by 10^x.

brnd=[lower balance@raise
      balance=(rkd&getfrac<calcrgap)+<.
      calcrgap=.swingu@((+/-) - +/@<.)
      getfrac=.1&|
      rkd=./:@\
      NB. balanced rounding
      NB. adjust y.to nrst integer
      NB. calculate rounding gap
      NB. get fractional part of y.
      NB. rank y. downwards
```

APL Berlin 2000 Conference Report



*notes by Ian Clark, Anthony and Sylvia Camacho
(collated by Stefano Lanzavechia). Pictures by Ray Cannon,
Dieter Lattermann, Anthony Camacho & Adrian Smith*

The conference took place in the centrally located Technical University of Berlin (TU). Delegates lodged in a number of hotels nearby (or in the Friends of Nature youth hostel not-so-nearby) and came in on foot or by Berlin's excellent public transport system of U-bahn, S-bahn and bus.



The hub of the conference was the vast marble-columnaded hall of the TU, which amply contained the conference desk, refreshments buffet (coffee and tea were permanently on tap), plenty of tables for consuming the same and the vendors' exhibitions.

Talks took place in two large auditoriums nearby, mostly in two parallel sessions in addition to the

series of workshops which took place in the Mathematics Building across the road, plus lunchtime vendor presentations by Causeway, Dyadic, IBM and Cognos.

Delegates were greeted on the first morning with handouts from the British APL Association (BAA), the latest issue of VECTOR and a CD with most of the conference papers in portable document format plus some giveaways. The BAA had offered a fair amount of behind-the-scenes support to make sure the conference was a success, but the German APL Association must be given credit for doing the real work. The organisers were pretty skilled at fitting a quart into a pint-pot - in spite of a full programme I never had the feeling of being hurried, whether over meals and coffee-breaks, or getting from one place to another.



A nice touch was to put out piles of copies of the papers currently taking place, or hand these out in the auditorium itself, so you had the paper to refer to whilst listening to the talks (alas, not every speaker's visuals were legible from the back of the auditorium.) The organising committee was always on-hand to help out with problems, and it seemed as though wherever I looked I could see Dieter and Dominika, busy as bees, slipping between the chattering groups sorting out some problem or other.

There was a distinct difference in culture between the presentations from former 'western' and 'eastern bloc' countries. The former were generally entertaining and made use of good quality visuals. The latter, alas, added little as a rule to the written paper which was already in the hands of the delegates in both printed and CD form. Often the text of the paper itself was what was projected onto the screen. There are probably several extenuating reasons for this, which may disappear in a few years as the former 'east' catches up, so to speak, in both quality and availability of resources. One cultural reason might be that speakers viewed the presentation as an onerous formality, the value of the conference to them having been largely for the social and personal contacts it afforded. Accordingly the objective of presenting their results to their peers had been met the instant their paper had appeared in the proceedings.

These papers it must be said were generally interesting, informative and in good to excellent English, a language in which the speakers themselves may not have had much practice in delivering presentations. Certainly nowhere near as much as the 'westerners', who were of course using their native language. There may also once upon a time have been a distinct disincentive to making good presentations by western standards: anything too easy to follow or suggestive of 'light entertainment' might have got your grant cut for trivialising the subject. One hopes that this reason has gone away by now and that the student graffiti on the auditorium desks which the reviewer had plenty of time to read ('langweilig' - boring - being a recurring word) are simply petroglyphs from a dead culture.



Back to the conference itself. As well as an extensive programme of cultural visits for spouses and other non-participants accompanying the delegates, there was something happening every night. Monday evening we gathered from the four quarters in the Cafe Filmbühne situated in nearby Steinplatz and there was much hailing of old acquaintances.

Tuesday evening was warm, if cloudy, and we took a trip along Berlin's river Spree on one of the original 19th century riverboats, the 'Fritz Zille', enjoying a buffet supper washed down with plenty of beer (when it could at last be coaxed out of the cask) to the sounds of a barrel-organ capably cranked by the skipper.



Adrian and Carlo competed with their digital cameras to capture the Most Gorgeous Nose of APL 2000, whereas the accolade for the most gorgeous pair of legs must surely go to the Belle of the Ball, Alexei's lovely daughter Masha, who stood out stunningly against the sombre ranks of us APL gnomes, unchained from our desks for a few mad summer days. (Sorry Lynne - but you did win the Ken Iverson Award for services to SigAPL!)

On Wednesday we boarded coaches for a trip to the ancient Citadel of Spandau, where we were treated to a mediaeval banquet plus a cabaret of horned pagan dancers with drums and German bagpipes (playing old Scottish tunes - shades of the McGinsbergs, of that ilk). The meat came in huge hunks which we carved ourselves, washed down

with plenty of wine and great jugs of beer - they could have fed another two beanfeasts on our leftovers!



If ample good food and good organisation are what is needed to make a conference go well, then APL-2000-Berlin couldn't fail to succeed. When the conference wound-up on Thursday afternoon it was agreed by one and all that we'd had an excellent time, made several new friends and even learned a few APL tricks we might not have thought of, left to ourselves. [Ian Clark]

[Editor's note: what follows is a short summary of a selection of papers and talks presented at the conference as provided by Ian Clark unless otherwise indicated.]

Dieter Lattermann, Conference Chairman

P.-M. Hager: Representation of ASN.1 in APL Nested Structures



Peter-Michael Hager, looking resplendent in a blue and yellow tie-dyed silk shirt which nicely set off his vivid red hair, had overheads which were photocopied from his paper. These were hard to read on the screen from the back, but whilst the talk was underway the photocopied paper was distributed to all listeners, which helped immensely.

It made a nice keepsake, because it contained a fully worked example of the syntax and resulting sample APL nested structure of a signed X.509-standard certificate for net transmission, serving as a handy introduction to the syntax and usage of ASN.1 (Abstract Syntax Notation One). This, Peter-Michael asserted,

was the standard for OSI (Open Systems Interconnection) and was used by several TCP/IP protocols and security applications.

After presenting this worked example in detail, a good one for its varied content, which included name and address fields, the speaker finished by offering an ASN.1 syntax for an entire APL workspace in platform-independent form, something else one might well want to transmit down the wires from time to time.

M. Barghoorn: High Performance Computing through Parallel Processing

Martin came to this problem through having to write network tools for the extensive university system. He demonstrated a program which set up and managed a collection of shared variables using {each} and {execute-each}, thereby achieving a measure of parallel processing. This shortened the time in which, say, the disk-space usage of all workstations could be determined across the network.

J. Brown: What's Wrong with APL2



Jim Brown (after 31 years with IBM, now of Smart Arrays) told us that he may have deceived us by his title of "What's wrong with APL2" into thinking that there was something wrong with it. What he actually did was recall some places where a different design decision would have been equally good or maybe even better and some places where a feature could have been added with advantage.

If Jim could change two things in hindsight he would like (1) to have created the function library that Gary Bergquist is working on now and (2) to have called it , not "APL2", but either APL version 2 (which would have persuaded more people to migrate) or by a completely new name - such as "Java". [A. Camacho]

R.G. Brown: Defining APL Community: Case Studies, towards a Revival of APL Community

The speaker, scorning hi-tech visuals by employing handwritten transparencies, gave two interesting potted histories of computer 'communities', which he defined as a group of people whose job it is to use the same sort of software. The idea was to discuss one 'dead' one, the LISP community (which died along with its main focus area, Artificial Intelligence), and one currently thriving one, the LINUX community, of which he is a leading light of a local group in Kalamazoo. These suggested a prescription of what might work to popularise APL better and to avoid the LISP pitfalls.

S.J. Halasz: An Improved Method for Creating Dynamic Web Forms Using APL

Steven described in broad terms work mainly done by Lingo Allegro for a Swiss customer, an existing APL user, to demonstrate the generation of dynamic web forms for their intranet. That is, the HTML defining the form is dynamically generated by the server, not written in fixed form. He described the advantages of various different approaches, discussing how (say) Cold Fusion and ISAP generate non-standard tags intended to be preprocessed, not to work the browser directly. A 'visual' editor such as Dreamweaver does not recognise these, so humble Notepad behaves better in practice. He argued for a 'thin client' rather

than a 'fat client' which takes time to download a fair-sized lump of code on first access which the client needs to show the page.

His preferred solution to the customer's problem, which he called a Document Oriented Interface (DOI), retained part-complete, unsubmitted user input forms across sessions, maybe containing errors, rather than a transaction approach which would annoyingly discard user input if the session did not result in a complete and correct form being submitted. It relied on secure (re-)identification of the user to do this, which led to some discussion about achieving this.

T. Laurmaa: Avoiding the Pitfalls of Corporate Intranets



Timo began with a reminder that he often presents light-hearted papers but that the audience could know by the fact that he is wearing a tie that, on this occasion, the topic is serious! By intranet he means a corporate-wide network constituting a group with read write and delete access to corporate data and with both rights and responsibilities with reference to the data.

A typical and often neglected responsibility is to *clean-up* the data they post when it becomes redundant. Also important, in the corporate context, is that they need a search engine that can answer the question "what should I know about today?" So, rather than create links to

many documents from within one HTML page, we want to create links to one document from many HTML pages. The best person to look after the timeliness of data is the creator himself at the time of creation. It should therefore be arranged to assign life-cycle controls which will remove out of date documents and, just as important, out of date links. The solution described is based upon Dyalog APL/W. [S. Camacho]

H. Ehrbar: Graph-theoretical Notation for Higher Dimensional Arrays

Hans Ehrbar is a mathematician turned economist who has developed a pictorial notation for arrays and would like to see its use extended. An array is shown as a tile with a number of arms corresponding to its rank. Joining tiles by connecting an arm of one to an arm of the other produces a new representation with the

joined arms excluded from the dimensionality. He showed how to use the notation to represent Einstein's summation convention, Hadamard and Kronecker products and how to handle commutation matrices and matrix differentiation. [A. Camacho]

A. Karabanov: Dyalog-APL Application with Threads on the Base of ActiveX Data Objects

The talk was re-titled: 'Use of multithreading features of Dyalog APL for development of database applications'. Co-author Günther Roche gave the talk in a clear voice, with well-designed Powerpoint slides, surviving (with session leader Peter Donnelly's help) not a little aggro from the Russian contingent, who were not convinced that his demonstration really did exhibit multi-threading, or its advantages if it did.

But the aim of the two main experimental techniques were clear and simple, to avoid the database accessing program waiting in an idle state whilst the database processed its enquiry. To do this they relied upon Dyalog APL's multi-threading capability to permit the enquiring algorithm to continue asynchronously, doing useful work, pending the reply from the database. Thus, 'useful work' might be to process the result of the previous enquiry. Various strategies of slicing up the problem, e.g. by batching up enquiries, were under investigation.

Günther actually demonstrated two examples of an APL program communicating via ADO with a database, employing Dyalog's approach to ADO (ActiveX Data Objects), which Microsoft now recommends in preference to OLE for all purposes. This approach generates the 3 kinds of data objects specified by ADO, 'Command', 'Connection' and 'Recordset', by applying these words as left ('type') arguments to a `⊞WG` statement, thereby generating the sets as namespaces. The reviewer detects a welcome trend to converge the syntax of creating and updating Dyalog namespaces and GUI objects, in spite of their differing origins. Whence might this lead? To being able to assign any nameclass 9 object to an APL identifier? To an ever-increasing overlap between the functionality of `⊞ns` and `⊞wc/⊞ws/⊞wg`? Even maybe to introducing a new migration level (oh, no! - not another `⊞ML` value!) at which GUI attributes become variables in the namespace (shades of Visual Basic!)? Could you then assign the values of some attributes of GUI1 to GUI2 and not others by an APL-like syntax instead of some clumsy nesting of `⊞WG` in `⊞WS`? Clearly this reviewer is letting his imagination run riot, as the denizens of Basingstoke will hasten to agree. [Editor's note: actually the reviewer is being prophetic since Dyalog APL version 9 provides exactly the kind of functionality imagined.]

R.L.W. Brown: Interest Made Simple with Arrays

Richard Brown (of York University, Toronto) showed that there are many problems in simple interest which become really tedious if you try to solve them with a calculator. He went through several examples of increasing complexity and showed how concisely they could be solved by simple expressions in J. He suggested that we wouldn't really appreciate the benefits of his methods unless we have tried to do the examples on a financial calculator and on a standard spreadsheet. Some of the things he did would be really difficult to do without the J facility for performing the inverse of a function. Then he showed how, by a simple substitution, the simple interest formula could be replaced by the compound interest formula in all the examples. [A. Camacho]

D. Sengenleitner: GERVA (Secure Electronic Legal Communication with Attributes)

GERVA provides IT services for lawyers and accountants specialising in tax affairs and their clients. Hitherto lack of security has meant that users hesitate to send legal documents by email. Dietmar commenced with a cartoon captioned: 'On the internet nobody knows you're a dog'. He reviewed European law on digital signatures, and described DATEV's Trust Centre, which issues the smart-cards and PINs. With a smartcard you can sign a document and encrypt it to be read only by selected subscribers. The smartcard is inserted into a reader connected to a Windows 9x or NT system via the serial port.

DATEV's system, GERVA (*Gesicherter Elektronischer RechtsVerkehr mit Attributen* - Secure Electronic Legal Communication with Attributes) is implemented in Dyalog APL. It is being field-tested with 580 participants from several organisations. Dietmar demonstrated the system in use with MS Outlook.

During questions from the floor it appeared that GERVA complies with European standards, rather than the de facto 'standard' shared by four USA security packages. Like standard cash-cards there seems to be no provision in DATEV yet to warn against use of the smartcard under duress. The common way to protect against this is to have not one but two passwords, the second password appearing to behave exactly like the normal password but signalling a covert warning at the control centre (the DATEV Trust Centre it would have to be in this case.) 'Common' though it is, this reviewer is not convinced of its psychology. If suitably blood-curdling threats were employed, would even the coolest user under duress really dare give the alarm password? It sounds like one of those features put in to fulfil a popular checklist of security features, rather than one

which would stand being put to the test (remember what happened to the over-heroic detective in 'The Firm').

M. Guazzo (discussion): APL and On Line Analytical Processing (OLAP)

Mauro Guazzo led this discussion about OLAP, which he defined as: response to unanticipated requests in a short time. The term OLAP, he said, came in around 1993 as a replacement term for Decision Support Systems (DSS). APL offers hypercubes to the problem, but he was not convinced that APL was the whole solution. In practice he himself used APL with customers only as a customisation tool of last resort. Could the APL community cooperate in identifying and researching the problems? – he asked.

He invited participants to identify themselves if they had undertaken significant work in OLAP and to say what for them were the chief problems. Which they did (most of the erstwhile Adaytum developers were there!) He also asked where current APLs were considered to fall down over OLAP. This is the gist of the main issues raised.

- Data model is some sort of hypercube, but a sparse one, or even one splitting into disjoint subcubes.
- Trade-off between flexibility and protection (of the user against himself).
- Missing values, or values too small to show. How infinitesimals can propagate, leading to unequal cells containing zero (effectively). $\square CT$ is not the whole answer.
- Representation of time-series, and totalling/aggregation.
- Relational approach is only procedural – when you get nearer the application you need OLAP.
- Relational approach falls down with time-series.
- Dimensions need classifying, e.g. the time dimension.
- Time series inherently requires aggregation (and in conflicting ways: e.g. summing over weeks, months, tax-years). Is this any different from, say, the nesting of regions? – argued Mauro. The reviewer thinks it is, because of the inherent time-ordering whatever split into time-intervals happens to be chosen, e.g. days, weeks, months, quarters, or Western dates vs. the Islamic lunar-based calendar. A 'regional' model in which relationships like 'within' 'to the east of' and 'overlaps with' are to be represented, might just begin to exhibit some of the special problems encountered with time dimensions.

- Because of the extensive need to flag array cells in one way or another, leading to a proliferation of Boolean arrays, many of which need saving across sessions, APL needs effective sparse Boolean matrix handling. J has got it, but it generally needs to be smart (and problem-specific) to be successful.

T. Otto: An APL Compiler

Using some refreshingly clear Powerpoint slides of APL and C examples, Tilman described the detail of his APL compiler (called APL2C) for generating C code from APL. The approach he chose was to write an entire APL language processor in ISO standard C, described as APL2-like, with a system function APL2C which generates C source. This allows APL code to call C functions directly and vice versa.

The reason he wrote it might be described as socio-organisational. The organisation was only happy with C code, but he wanted to solve his medical imaging problems in APL (see below). The solution? Simply write your own APL interpreter in C. Hats off to him! The result could be more generally useful for the APL expert/software developer but, alas, the compiler is not for sale. One nice feature, a bonus of writing your own APL interpreter, is that you can put the APL symbols on the session manager toolbar (or even just the ones you have most difficulty finding on the keyboard). Such an obvious trick in hindsight.

Sample tests: Complex visual image processor, 106 fns, 1690 lines incl. comments → 28,000 lines of C (with embedded comments). Performance tests show that the savings by converting the application from APL to C are attributable just to the APL interpretation overhead.

T. Otto: APL Based Medical Image Analysis

Tilman used his own APL2C interpreter/code generator (which he had already presented in a separate paper, see above) to perform filtering, averaging and animation on medical images of the human retina. He showed some impressive slides of the result.

A. Skomorokhov: One Knowledge Discovery Method. APL Implementation and Application

Before the speaker began, the projector showed his desktop: a photo of a supine cat. He was asked from the audience if this was Schroedinger's Cat, but dodged the implicit issue of whether it was dead or just sleeping.

Abandoning his pixillated Powerpoint slides, he reverted to a WORD image of his paper. This offered an algorithm to infer 'rules' (complex logical predicates) from repeated tests on data, using a simple table of (non-nuclear) sample data as an example (sex, income, colour). Problems with the APL font in WORD led to him loading the workspace and attempting to run the example by hand.

The speaker had applied his methods to investigating the corrosion rates of various kinds of steel in liquid sodium for the nuclear industry, alerting the engineers to rule parameters they had not considered. The implied agenda is of course to avoid a follow-on from the people who brought you Chernobyl.

A.D. Mayer & A.M. Sykes: GrAPL - A High-Level Statistical Graphics Prototype Language



In a sense, this was yet-another RainPro interface, but one which the speakers from the University of Wales made a good case for being needed in academic statistical investigations. The existing RainPro interface can be described as 'GDDM-like' - a series of separate calls to the package sets up the desired graph and a final one gives the command to draw.

GrAPL crams all the 'relevant' parameterising information into a single APL call. The trick of course is to do this without weighing down the syntax with parameters you'd rather not have to specify at this stage. The reviewer has seen a long succession of imaginative solutions to this problem ever since he worked on improving the unsatisfactory human factors of the GDDM Chart Utility when it first came out in 1980 or thereabouts. To say that the last word on this perennial topic has yet to be spoken is not to detract from the authors' achievement. Like many an interface to an advanced general-purpose product, there may be no 'ideal' syntax to support all the product's tasks in all environments - it's a case of 'horses for courses'. What excited this reviewer was how the speaker clearly showed his horse jumping its fences.

M. Alfonseca: Artificial Life Evolution in a Simplified APL2 Environment



The whole topic of Artificial Life is a bit like 'crystallised humanity' to Mephistopheles in Goethe's *Faust*. The longer you live the more you feel you've seen it all before. However this does less than justice to the author's paper. The purpose of model-making on this tiny scale is to exhibit interesting phenomena in the classroom to students who might expect it only in living systems, with all their overwhelming complexity.

This is one more example to add to the burgeoning field of Genetic Algorithms, in which an 'organism' is improved (i.e. the task it performs is optimised) by breeding it with variations (or mutations). The organism can take a variety of forms, e.g. one early investigator actually used a Von Neumann Games Theory payoff matrix for the game of Poker, simplified of course, 'bred' the organisms in an elementary Mendelian sexual way and played the results off against each other at each generation. After 12 or so generations, the 'optimal' payoff matrix evolved (Games Theory of course uses the minimax process for deriving the optimal matrix in a wholly different way.) The choice of task for the organism was original, the sex-like mechanism wasn't. Nor was it intended to be, because the choice of breeding mechanism was seen just as a way of converging faster on the optimum solution.

In the speaker's case the organism was a midget program in a 'toy' machine-code specialised to the purpose of self-replication. So the task to be optimised and the breeding process were one and the same, which seems to be the main claim to originality. One might then expect completely novel breeding processes to arise, which is what happened. Another thing that happened was that a given 'type' of organism appeared, went extinct and then re-appeared. The reviewer suspects that this is an artifact of the tiny scale of the model, which reduces the organism's variety (in the Ross Ashby sense). The reviewer doesn't believe in the slightest, for instance, that if humanity became extinct it would essentially re-evolve given sufficient time. ('It's humanity, Jim, but not as we know it!')

E. Juvonen : The Making of a Conference – APL 92 in St Petersburg, Russia

Erkki Juvonen's account was fascinating. He chose to hang the presentation on an idiosyncratic selection of the places where the project could have been torpedoed. (For example the janitor of the Danish Embassy in Moscow might not have helped to get the missing signatures on Andrei and Olga Kondrashev's visas.) He arranged these critical events as the lines in a function, numbered accordingly. [A. Camacho]



D. Baronet: Introduction to Tree Searching

Daniel Baronet presented, in a model of clear concise exposition, an introduction to trees and tree searching. The motto was (1) check for solutions and generate new branches (2) search each one of them (recursive process). He then followed this advice to generate permutations of three letters and then to generate permutations of the sums of numbers (such as one has to do when allocating cash to invoice items) and in this example showed how to reduce the number of permutations to be searched. He then used the same techniques to solve the eight queens problem and the knight's tour problem. These last two examples enabled him to show how careful one must be to choose an approach that doesn't do unnecessary work. Finally he showed a program for three-dimensional noughts and crosses (tic tac toe) which he said had not been beaten if allowed to take the first move. Some of the programs are on the CD – the 3d tic tac toe in versions for Dyalog, APL+Win, APL*PLUS/PC and J. They look well worth exploring. [A. Camacho]

A. Smith: Redistribution of Totals through Hierarchical Data

Adrian Smith raised the problem that, in some circumstances such as budgeting, one wants to be able to alter a total and have the alteration spread among the items adding to the total so that the sum exactly equals the new total. If you want to reduce the vector $4\ 3\ 2\ 5$ (=14) to add to 13 then multiplying by thirteen

fourteenths and rounding will leave everything unchanged. Phil Benkard solved the problem in his paper 'Dance of the rounds' given at Stanford in APL 91. The method that Adrian recommended was to apply the prorating to the plus-scan of the items and then take the first difference of the prorated scan. The result is always correct, it is very fast and easily extends to higher dimensions.

Then Adrian raised the problem of updating the annual budget three months into the year. The first three columns are now actuals and cannot be changed so the prorating has to be done on the remaining nine. This is how the budgets for the later parts of the planning year can get to be more and more unrealistic as the year goes by. The example function he gave can easily be amended to 'hold' any of the data and forces any changes to be spread among the data not 'held'. A budget is often held as a table or higher rank array in which the row, column, plane &c totals are significant. In Adrian's system it makes a difference to the result whether one rounds across or down or throughout the ravelled table.

Questioners suggested that there were, in some cases, better and worse rules for allocating the items to be increased or decreased above or below what the simple rounding would achieve. In particular it may be better, if one of the values with a prorated fractional part of less than .5 has to be rounded up then it should be the largest one (the error in that value will be a smaller proportion of the value) or perhaps the value with the largest of the sub .5 fractions should be the one to round up. Adrian admitted that the method he had chosen was, in effect, a random choice of item to be increased or decreased yet maintained that the advantages of speed and simplicity made his choice the best. [A. Camacho]

M. Symes: An Interface between Java and APL

Mike gave a brief review of the similarities between Java and APL, for instance both are interpreted and both have an execute string instruction which allows each to initiate actions in the other. On the other hand Java uses very strong typing and APL is almost type free while Java, unlike APL, has no scalar extension. This is a very 'meaty' paper with many examples of interface code and descriptions of the various technical issues to be addressed. [S. Camacho]

R. Brown: J in Your Pocket

Richard Brown demonstrated a Casio E115 palmtop computer running CE on which J had been implemented. Entry was by means of a stylus with which you pressed the letters on a picture of a keyboard on the lower part of the screen. Martin Nietzel did the first port. The basic machine has a rather limited memory but you can add flash cards up to 64 megabytes which, although slower than

memory seem as fast as a hard disk. The new version of Windows CE is called PocketPC. The professional version (non-commercial use) of J is free for this kind of device and the screen has a high enough resolution to display quite complex graphs. As Eric Iverson says, "If you need a really serious pocket calculator there's nothing to touch it". NEC, Sharp and HP also make palm top computers with similar facilities. [A. Camacho]

I. Clark: The 'Killer App': How to Make Millions with an APL Product



Another plenary session was the highlight of the conference: Ian Clark told us how to make millions with an APL product. His talk was largely about how to choose the *Killer Application* with hints on how to avoid numerous pitfalls in the exploitation of an idea. The *Killer App* took the fancy of the delegates and the idea appeared in many of the later presentations. Ian produced more epigrams in fifty minutes than the rest of the conference could manage in a day. The audience went away laughing, stimulated and happy in the thought of the riches to come. [A. Camacho]

The Workshops and Tutorials [Adrian Smith]

I suppose the workshops are the hardest thing to set up, and in some ways were the least successful of the mainstream events in Berlin. They were held in the Maths building, as this was the nearest place where there was access to a room full of computers. In fact the computers were so thoroughly student-proofed that there was no possibility of installing an APL font on any of the desktops, so all the sessions were actually 'talk and chalk' mode rather than true hands-on events.

The other problem with the Maths building was that it lay across a busy main road, and once inside it was no trivial challenge to navigate to the correct room ...

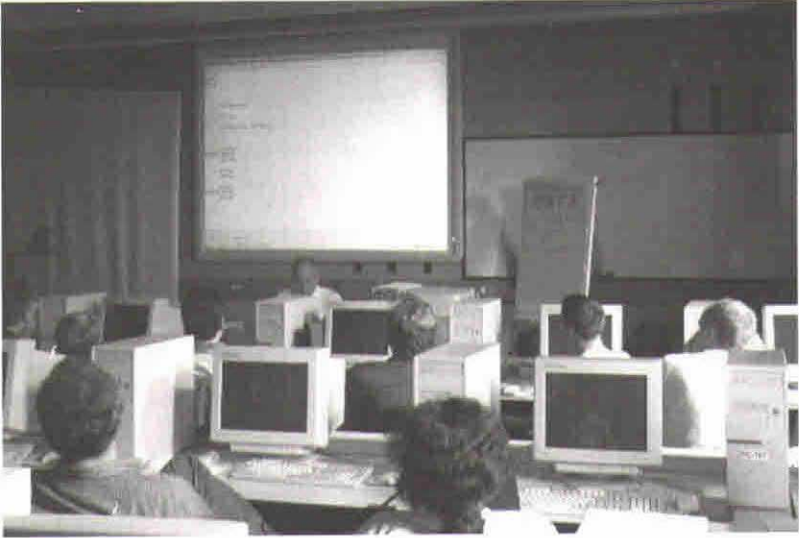


... although the architecture was pretty impressive along the way! The main attraction was again the run of Dyadic workshops (you can read Pete Donnelly's

notes in full on the following pages) which were well over-subscribed, as you can see here. This also shows that the computers were simply an impediment, and were mostly turned off to keep the heat and noise down. This also deterred all but the most anti-social delegates from using the



session as an opportunity to check and answer emails! Unfortunately the computers were more than capable of surfing the internet, which is something else that organisers of future conferences should look out for.



John Scholes, but not a Duck in Sight!

Having said all that, I should finish with the comment that the Dyadic seminars were very well presented, the material was interesting, and the audience as attentive as you can be in a hot dark room after a good lunch.



... and the reciprocal of π to your dear wife ...

Driving the Windows GUI and COM from Dyalog APL Version 9.0

workshop notes by Pete Donnelly (Dyadic Systems)

This tutorial explores the manner in which Dyalog APL Version 9 has been enhanced to make access to the Windows GUI and COM easier to program and faster to use.

Let's start by making a Form:

In the session menu, there is an option labelled *Options/Object Syntax/Expose GUI Properties* which I have checked. This means that I can manipulate a property directly, as if it were a variable:

```
F.Caption
F.Caption+ 'Hello World'
F.Caption←F.Caption
F.Caption
dlroW olleH
```

If we change into the *F* namespace, we can use the new *)PROPS*, *)EVENTS* and *)METHODS* system commands to discover the names of the Properties, Events and Methods provided by the object.

```
)CS F
#.F

)PROPS
Accelerator AcceptFiles Active AlphaBlend AutoConf
Border BCol Caption ChildList Coord CursorObj
Data Dockable Docked DockChildren DockShowCaption
EdgeStyle Event EventList FontObj Handle HelpButton
Hint HintObj HScroll IconObj KeepOnClose MaskCol MaxButton
MethodList MinButton Moveable OnTop Picture
Posn PropList Range Size Sizeable State
Step SysMenu TextSize Thumb Tip TipObj Translate
Type UndocksToRoot Visible VScroll XRange YRange
```

```

)EVENTS
Close   Configure   ContextMenu   Create   DockAccept
DockCancel   DockEnd   DockMove   DockRequest   DockStart
DragDrop   DropFiles   DropObjects
DialogCustomMessage1   Expose   FontCancel   FontOK
FrameContextMenu   GotFocus   Help   HScroll   KeyPress
LostFocus   MouseDblClick   MouseDown   MouseEnter
MouseLeave   MouseMove   MouseUp   MouseWheel   Select
StateChange   Vscroll

)METHODS
Animate   ChooseFont   Detach   GetFocus   GetTextSize
Wait

```

Once in the *F* namespace, we can of course omit the name of the namespace itself when we refer to a property. For example:

```

BCol
0
BCol+255 0 0

```

There is a new *AlphaBlend* property (Windows 2000 only) that controls the translucency of a Form. For example:

```

AlphaBlend+128
AlphaBlend+256

```

There is a new *Animate* method (Windows 2000 only) which shows and hides an object with an animated effect. For example:

```

Animate 16
1

```

The optional second parameter specifies the play time:

```

Animate 16 2000
1

```

As we are in the Form namespace *F*, we can create a Grid object named *G* as follows:

```
'G'[]WC'Grid'
```

Again, we can manipulate the properties of the Grid directly.

```

G.Posn
24.82014465 25

```

Indexing works as you would expect:

```
G.Posn[1]+40
```

As does Strand (vector) assignment:

```
G.Posn G.Size+(10 10)(80 80)
```

You can also manipulate several properties together using an expression such as:

```
G.(Values CellWidths)+(710 10p100) 10
```

Incidentally, the `Animate` method also works on the Grid (and on most other objects) but the animation only applies when making it visible:

```
G.Animate 16
1
G.Animate 16
1
```

The new syntax is especially convenient when combined with `:With`. Here is a function that illustrates this nicely.

```

)CS
DUCK
v DUCK;F;SINK;ANIM
[1] ANIM+1 2 4 8 16 a Animation styles
[2] :Trap 6
[3] 'F'[]WC'Form' 'Duck à la Grid'
[4] :With 'F.G'[]WC'Grid'
[5] Values+50 50p<'
[6] Posn Size+(0 0)(50 50)
[7] CellWidths CellHeights+6
[8] TitleWidth TitleHeight+0
[9] 'D'[]WC'Poly'#.GDUCK('Coord' 'Cell')('FStyle' 0)
[10] :While 1
[11] Visible+0
[12] Posn+?[0.5*##.Size
[13] BCol+<?255 255 255
[14] D.FillCol+?255 255 255
[15] SINK+Animate(=ANIM),1000
[16] ANIM+1φANIM
[17] []DL 0.5
[18] :EndWhile
[19] :EndWith
[20] :EndTrap
v


```

Let's now take a look at the Event property.

```

F'[]WC'Form'
)COPY DISPLAY
DISPLAY F.Event

```

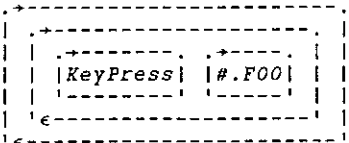


You can assign to the Event property:

```

F.Event←'KeyPress' 'FOO'
DISPLAY F.Event

```

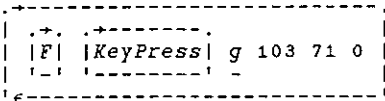


```

▽ FOO M
[1] DISPLAY M
▽

```

Now, if I press a key on the Form, the callback fires and displays the event message:



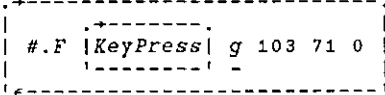
As you can see, the first element of the event message is a character vector containing the *name* of the object that caused the event. This is compatible with previous versions of Dyalog APL.

If however, we prefix the event name with *on*, the first element of the event message is a *namespace reference* to the object

```

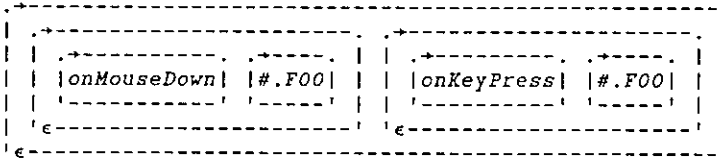
F.Event←'onKeyPress' 'FOO'

```



To make it easier to assign events, you can assign to `on<event name>` directly:

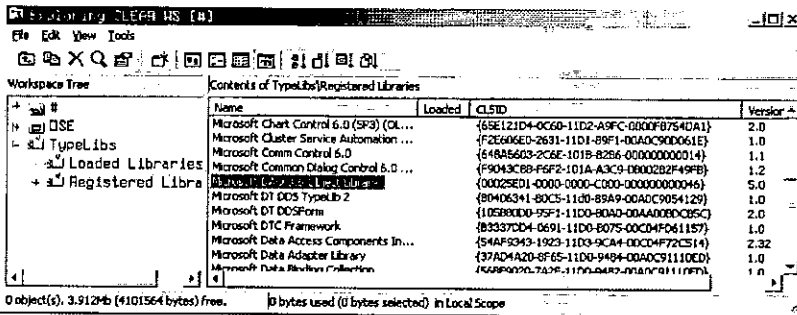
```
F.onKeyPress+'FOO'
F.onMouseDown+'FOO'
DISPLAY F.Event
```



Microsoft Jet Engine

We will start our exploration of the Version 9 COM interface, using the DAO.DBEngine OLE Server (also called the Microsoft Jet Engine).

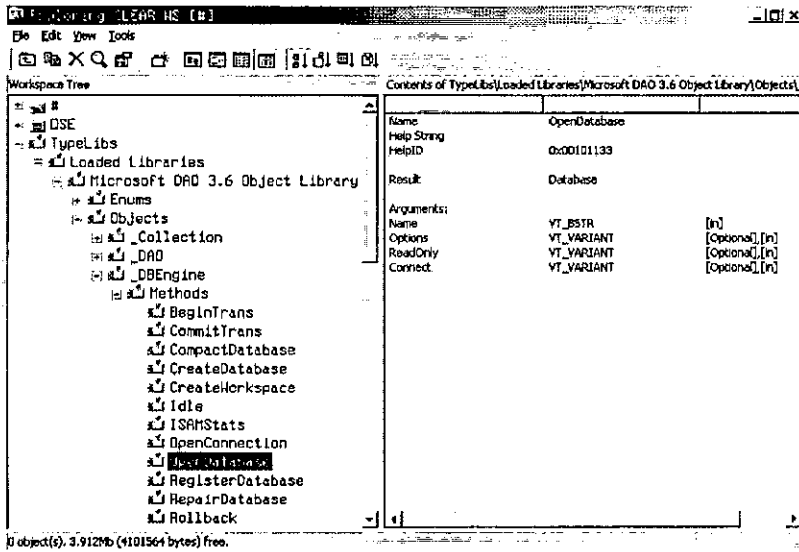
Let's begin by using the new COM Browser that is built into the Version 9 Workspace Explorer.



The *Registered Libraries* folder contains information about all of the libraries associated with COM objects that are installed on your computer. You can navigate to the *Microsoft DAO 3.6 Object Library*, as shown above, and then double-click to load it.

The *Loaded Libraries* folder contains information about all of the libraries that are loaded into the active workspace. You can navigate to the `_DBEngine` object, and then to its `OpenDatabase` method, as shown below.

The Opendatabase method takes a single mandatory parameter, the name of a database file, and a number of additional optional parameters.



To save me typing, I am going to load a workspace:

```
)LOAD PDDEMO
```

Leaving the Workspace Explorer for now, we can create a new OLEClient namespace *DAO* connected to this server as follows:

First, I am going to set up `PATH` so that I can refer to any function in the root from any namespace:

```
PATH←'#'  
'DAO'←WC'OLEClient' 'DAO.DBEngine.36'
```

Then, change to the DAO namespace and look at the properties and methods that the object provides using the `)PROPS` and `)METHODS` system commands:

```

)CS DAO
)PROPS
AutoBrowse      ChildList      ClassID  ClassName      Data
DefaultPassword DefaultType  DefaultUser Errors  Event
EventList       Handle      HelpFile      IniPath InstanceMode
KeepOnClose     LastError   Locale  LoginTimeout  MethodList
Properties      PropList    QueueEvents  SystemDB
Type            TypeList   Version  Workspaces

)METHODS
BeginTrans      Browse      CommitTrans  CompactDatabase
CreateDatabase  CreateWorkspace Detach  GetEventInfo
GetMethodInfo   GetPropertyInfo GetTypeInfo   Idle      ISAMStats
OpenConnection OpenDatabase OLEAddEventSink OLEDeleteEventSink
OLEListEventSinks OLEQueryInterface RegisterDatabase
RepairDatabase Rollback     SetMethodInfo  SetOption
SetPropertyInfo ShowHelp

```

Position the cursor over the word `OpenDatabase`, and invoke the on-line help by pressing F1 or using the context menu.

You will notice that the on-line help confirms what we discovered previously using the Explorer.

To save typing, the `NORTHWIND` function returns the full pathname of the sample `Northwind.mdb` database on my computer:

```

NORTHWIND
c:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb

```

The `OpenDatabase` function returns an object, i.e. a namespace reference, which we can assign to a name `DB`:

```
DB←OpenDatabase NORTHWIND
```

If we ask what's in `DB`, APL tells us that it is a namespace reference to a Database object:

```

DB
#.DAO.[Database]

```

Let's now change into the *DB* namespace and see what we find:

```

)CS DB
#.DAO.[Database]
)METHODS
Browse Close CreateProperty CreateQueryDef CreateRelation
CreateTableDef Detach Execute GetEventInfo GetMethodInfo
GetPropertyInfo GetTypeInfo MakeReplica NewPassword
OpenRecordset OLEAddEventSink OLEDeleteEventSink
OLEListEventSinks OLEQueryInterface PopulatePartial
SetMethodInfo SetPropertyInfo ShowHelp Synchronize

```

Once again, let's use the context-sensitive help to examine the *OpenRecordset* method. You will see that it can be called with an SQL query. The method returns a *Recordset* object, which we will assign to the name *RCS*.

```
RCS←OpenRecordset'Select * from Customers'
```

Now change into the *RCS* namespace and look at the properties and methods it provides:

```

)CS RCS
#.DAO.[Database].[Recordset]
)METHODS
AddNew Browse Cancel CancelUpdate Clone Close
CopyQueryDef Delete Detach Edit FillCache FindFirst
FindLast FindNext FindPrevious GetEventInfo
GetMethodInfo GetPropertyInfo GetRows GetTypeInfo Move
MoveFirst MoveLast MoveNext MovePrevious
NextRecordset OpenRecordset OLEAddEventSink OLEDeleteEventSink
OLEListEventSinks OLEQueryInterface Requery Seek
SetMethodInfo SetPropertyInfo ShowHelp Update

```

Using the context-sensitive help, take a look at the *GetRows* method. This takes a single parameter that specifies the number of rows to retrieve. Let's try asking for 2:

```

GetRows 2
ALFKI ANATR
Alfreds Futterkiste Ana Trujillo Emparedados y helados
Maria Anders Ana Trujillo
Sales Representative Owner
Obere Str. 57 Avda. de la Constitución 2222
Berlin México D.F.

12209 05021
Germany Mexico
030-0074321 (5) 555-4729
030-0076545 (5) 555-3745

```

This delivers the first 2 rows from the table. If we do it again, we get the next two (rows 3 and 4).

```

    GetRows 2
ANTON                AROUT
Antonio Moreno Taquería  Around the Horn
Antonio Moreno        Thomas Hardy
Owner                Sales Representative
Mataderos 2312       120 Hanover Sq.
México D.F.         London

05023                WA1 1DP
Mexico               UK
(5) 555-3932        (171) 555-7788
                    (171) 555-6750

```

The MoveFirst method resets the pointer to the first row:

```

    MoveFirst
    GetRows 2
ALFKI                ANATR
Alfreds Futterkiste  Ana Trujillo Emparedados y helados
Maria Anders        Ana Trujillo
Sales Representative Owner
Obere Str. 57       Avda. de la Constitución 2222
Berlin              México D.F.

12209                05021
Germany             Mexico
030-0074321        (5) 555-4729
030-0076545        (5) 555-3745

```

To get all the data, we can just ask for a large number of rows:

```

    MoveFirst
    ρGetRows 9999
11 91

```

I will now show you how this entire query can be executed in a single statement

First, we need to change back to root:

```

)CS
#

```

If we execute the following expression, we get a namespace reference to a Database object:

```
(DAO.OpenDatabase NORTHWIND)
#.DAO.[Database]
```

There is nothing to prevent us from calling the OpenRecordset method in this object, even though the object itself has no name: This gives us a namespace reference to a Recordset object.

```
((DAO.OpenDatabase NORTHWIND).OpenRecordset 'Select * from
Products')
#.DAO.[Database].[Recordset]
Finally, we can run the GetRows method in this object, giving us:
DATA+>((DAO.OpenDatabase NORTHWIND).OpenRecordset 'Select * from
Products').GetRows 9999
ρDATA
77 10
```

Microsoft Excel

Let's now experiment with the Excel OLE Server, *Excel.Application*

We can create a new OLEClient namespace *EX* connected to this server as follows:

```
'EX' □WC'OLEClient' 'Excel.Application'
```

Next, we will change into the *EX* namespace and look at its Properties:

```
)CS EX
)PROPS
_Default      ActiveCell      ActiveChart      ActiveDialog
ActiveMenuBar  ActivePrinter  ActiveSheet      ActiveWindow
...
Width  Windows  WindowsForPens  WindowState  Workbooks
Worksheets      WorksheetFunction
```

Let's make Excel visible by setting its Visible property:

```
Visible+1
```

I seem to remember that the way into Excel is to start with the Workbooks property. To check this out, I can click on the word Workbooks (as displayed in the session window) and bring up the context-sensitive help, using the mouse or F1.

Following the on-line help, I see that it has an Add method that can be called with or without arguments - let's try it without (the default). Note that to call it "without arguments", we supply \emptyset as the argument.

```
Workbooks.Add  $\emptyset$ 
```

There is another likely looking property named ActiveSheet; let's check out its help topic.

So ActiveSheet returns a Worksheet object. Let's look at its Properties. This time, we will use its PropList property (a property added by Dyalog APL) rather than the system command)PROPS which you can only use inside the namespace in question.

```
ActiveSheet.PropList
Type Application Creator Parent CodeName _CodeName Index Name Next
OnDoubleClick OnSheetActivate OnSheetDeactivate PageSetup Previous
ProtectContents ProtectDrawingObjects ProtectionMode ProtectScenarios
Visible Shapes TransitionExpEval AutoFilterMode EnableCalculation Cells
CircularReference Columns ConsolidationFunction ConsolidationOptions
ConsolidationSources DisplayAutomaticPageBreaks EnableAutoFilter
EnableSelection EnableOutlining EnablePivotTable FilterMode Names
OnCalculate OnData OnEntry Outline Range Rows ScrollArea...
```

Let's try changing its Name property:

```
ActiveSheet.Name+'PETE'
```

The Range property looks interesting - let's check it out using the context-sensitive help:

So, Range is a property that takes 2 parameters that specify the addresses of the first and last cells. It returns a Range object which, we can see from the help file, has a Value property. We can set this as follows:

```
(ActiveSheet.Range 'A1' 'J10').Value+710 10p100
```

Notice that the result of the expression (ActiveSheet.Range 'A1' 'J10') is a Namespace Reference to an (unnamed) Range object. We can then reference its Value property directly.

The Range object has a Font property that returns a Font object. The Font object has a property called Size; let's try it:

```
(ActiveSheet.Range 'A1' 'J10').Font.Size+32
```

Notice that in this case, there are two sub-expressions that return objects; (*ActiveSheet.Range 'A1' 'J10'*) and (*ActiveSheet.Range 'A1' 'J10').Font*

Similarly, a Range object has a Borders property that returns a Borders object (actually a collection) that has a LineStyle property. Let's query it:

```
(ActiveSheet.Range 'A1' 'J10').Borders.LineStyle
~4142
```

The help file tells us that LineStyle may be set to one of a set of XlLineStyle enumerated constants. In Version 9, you can refer to these directly.

```
XlLineStyle
xlContinuous      1
xlDash             ~4115
xlDashDot          4
xlDashDotDot      5
xlDot              ~4118
xlDouble           ~4119
xlSlantDashDot    13
xlLineStyleNone   ~4142
```

So, the following expression may be used to set the grid lines to double lines.

```
(ActiveSheet.Range 'A1' 'J10').Borders.LineStyle+xlDouble
```

If you are going to refer to an object more than once, it is more efficient to assign it to a name (if you were programming in VBA, you would be advised to do this too)

```
CELLS←ActiveSheet.Range 'A1' 'J10'
FNT←CELLS.Font
FNT.Bold←1
```

If you want to specify an individual border in VBA, you can reference it by an expression such as

```
Borders(xlEdgeBottom).
```

This works because the Borders collection has a default method (Item). In Dyalog APL, you cannot use this syntax; you have to call the Item method explicitly.

So, to set the colour of the horizontal lines to red;

```
(CELLS.Borders.Item xlInsideHorizontal).Color+25610 0 255
(CELLS.Borders.Item xlInsideVertical).Color+25610 255 0
```

... and the vertical lines to green,

The reason that it is necessary to encode the RGB colour, is that the data type of the Color property is defined to be VT_VARIANT. If a data type is VT_COLOR, you can assign a 3-element RGB vector to it directly. Notice also that the RGB triplet is specified in the reverse order (blue, green, red).

Microsoft Word

Let's now look at driving Microsoft Word. I am going to use a real-life example; one that we use to maintain the cross-references in the Dyalog APL Object Reference. Here is the function we use:

```

v UPDATE;W;REL;F;T;I;NAME;TYPE;SINK;OML;LIST;OBJ
[1]  OML+3
[2]  OPATH+'+'
[3]  GET_EVENT_MAP
[4]  'W'⊂WC'OLECLIENT' 'Word.Application'
[5]  REL+W.Documents.Open'C:\MANUAL\OR901.DOC'
[6]  W.Visible←1
[7]  :With W.ActiveDocument.ActiveWindow.View
[8]      ShowAll+0
[9]      ShowHiddenText+0
[10] :EndWith
[11] REL.(ShowSpellingErrors ShowGrammaticalErrors)+0
[12] :With REL.Content
[13]     F←Find
[14]     F.Style←wdStyleHeadings5
[15]     :While F.Found∧F.Execute''
[16]         T←Text
[17]         NAME TYPE←2+(~Tε[AV[4 5 10]])←T
[18]         :If TYPE='Object'
[19]             SINK←Move wdParagraph 2
[20]             :For ITEM :In OBJECT_INFO NAME
[21]                 SINK←Move wdWord 3
[22]                 SINK←MoveEnd wdParagraph 1
[23]                 Select
[24]                     Text←ITEM
[25]                 SINK←Move wdParagraph 1
[26]             :EndFor
[27]             SINK←MoveEnd wdSection 1
[28]         :Else
[29]             LIST←('Property' 'Event' 'Method'\<TYPE)>#.Properties
[30]                 #.Events #.Methods
[30]             OBJ←MAKE_LIST((<<NAME)∘.eLIST)/#.Objects

```



```

[31]          SINK+Move wdParagraph 1
[32]          SINK+Move wdWord 3 n Applies to <tab> <word 4 = stuff>
[33]          SINK+MoveEnd wdParagraph 1
[34]          Select
[35]          Text+OBJ
[36]          SINK+MoveEnd wdSection 1
[37]          :EndIf
[38]          :EndWhile
[39]          :EndWith
[40]          REL.Close REL.wdSaveChanges
[41]          W.Quit 0

```

v

Let's trace it and see how it works.

- [1] Use maximum APL2 compatibility
- [2] Search up the namespace tree for functions.
- [3] Reads the lists of Objects, Properties, etc which are stored in a component file.
- [4] Creates an OLEClient object named *W* connected to the OLE Server *Word.Application*.
- [5] Executes the Open method of the Documents object to open the Dyalog APL Object Reference file OR901.DOC for editing.
- [6] Sets the Visible property (of the Word application). (This is done only for the sake of this presentation.)
- [7] When editing a Word document it is often important to ensure that the various user/document options are correctly set. For example, this function will only work if hidden text is hidden. Let's use the on-line help:

```
W.ShowHelp 0
```

Now I am going to search for **hiddentext*

The ShowHiddenText property of the View object is what we need. By clicking *See Also* we can see another related property ShowAll. If we look at Options/Tools/View, we can see the check boxes to which these properties refer. The example illustrates how we can get to the View object via ActiveDocument.ActiveWindow. The :With control structure is a handy way (but not the only way) to set both properties:

- [8] If I right-click on the name ShowAll, then take a look at Properties/Value, I can see that ShowAll is set to 1. This line sets it to 0.
- [9] If I switch to the session and type

ShowHiddenText

1

This line sets ShowHiddenText to 0

- [11] Although this step is not strictly necessary (because, in production, the Word application is not visible), this line disables spelling and grammar checking (otherwise, I will get an unfriendly message box telling me that there are too many to cope with!). This could be done using *:With* as before, but an alternative is to parenthesise the strand assignment within the namespace reference. Before we execute this line, let's take a look at the current values of these properties:

```
REL.(ShowSpellingErrors ShowGrammaticalErrors)
```

1 1

- [12] Before executing this line, I will right-click on the text *REL.Content* and bring up the on-line help. As you can see, the Content property returns a Range object that represents the entire content of the document. The *:With* takes us into this (unnamed) namespace.

After executing the line, we can see the extent of the Range object from its Start and End properties. These indicate the starting and ending character index of the Range.

- [13] We are going to use Find to locate paragraphs with style *Heading 5*. Once again, I will bring up the on-line help for the Find property; it returns a Find object which is assigned the name *F*.
- [14] This line sets the Style property of the Find object to *Heading 5*. This is one of the standard styles, so we can use the enumerated constant *wdHeading5* to refer to it.
- [15] This line invokes the Execute method (which returns 1 for success) and then checks the Found property (belt and braces). The function loops until it runs out of *Heading 5* paragraphs.
- [16] If the Execute method of the Find object finds a hit, it resets the extent of its parent object (in this case, the unnamed Range object that we are *in*) to the Find target. Here's where we are in the document:

Abort

Method 103

Applies to Printer

This method causes the current print job to be aborted and all pending output to be discarded.

The Abort method is niladic.

If you attach a callback function to this event and have it return a value of 0, the print job will continue.

The Text property reports the text.

Let's look at the Start and End properties again.

```

      Text
Abort<tab>      Method 103

```

[17] At last, some APL code! This line splits out the name and type of the item found. *TYPE* will be *Object*, *Property*, *Method* or *Event*. The first item located is *Abort Method*, so we go to [29].

[29]

[30] These lines extract the list of objects to which the particular Property, Method or Event in question (*NAME*) applies.

[31] Using the on-line help, we can see that the Move method collapses the Range and moves the Start and End pointers the specified number of units. In this case, we move the Range (the Range we are *in*) forward one paragraph.

After executing [31], let's look at the Start and End properties again:

```

      Start End
88990 88990

```

[32] The Range is now positioned before the paragraph that starts *Applies to<tab>*. This line moves the Range forward 3 words, i.e. to just after the *<tab>*.

After executing [32], let's look at the Start and End properties again:

```

      Start End
89001 89001

```

[33] Using the on-line help again, we see that the MoveEnd method moves the *ending character position* of a range. This line moves the End character to the end of the paragraph, leaving the Start character unchanged.

```

      Start End
89001 89009

```

- [34] This statement selects and highlights the current Range. This line is not part of the production code, but I have included it to force the display to show where we are in the document.
- [35] This statement replaces the existing text (within the current Range) with the contents of *OBJ*. Notice that *OBJ* contains a new line character.

```

OBJ
Printer

```

```

□AV;OBJ
81 35 26 31 37 22 35 4

```

- [36] Having made the text replacement, we are ready to move on to the next item. However, don't forget what how we have restricted the Range we are in. This line moves the End pointer to the end of the section (i.e. to the end of the chapter we are working on).
- [37]
- [38] The code branches back to the start of the While loop to look for the next item with Paragraph style Heading 5. We have seen how Properties are dealt with, and Methods/Events are the same. Let's stop it after it finds an Object by putting a stop on [19], then let execution go on.
- [19] Rather than bore you with the code, which is essentially the same as before, I will just step through it and you can see what happens.

Finally, here is a little curiosity that I stumbled across.

The Word Application object has a method called *GetSpellingSuggestions* that returns a *SpellingSuggestions* collection. Each item in the *SpellingSuggestions* collection is a *SpellingSuggestion* object whose Name property contains the suggested word (pew!).

Using Dyalog APL, we can get suggestions for the spelling mistake "Hellao" as follows:

```

WORDS+W.GetSpellingSuggestions'Hellao'
{{(WORDS.Item w).Name}¨\WORDS.Count
Hello Hellion Hellas Halloo Hellos Hallo

```

R.I.P. OOF

(At the Hand of Dyalog v9.0)

by Morten Kromberg (mkromberg@insight.dk)

Introduction

In July 2000, Vector published an article I had written in April, describing a toolkit called "OOF" (OLE Object Functions), which I had developed in order to make it easier to use ActiveX or OLE Servers, such as those provided by Microsoft Excel. In this article, I concluded:

Dyalog APL provides most of what you need to use OLE Servers from APL, but in my opinion, they could be significantly easier to use. The OOF tool allows APL developers to use expressions, which are closer to that which they can find in documentation and other literature, which is usually aimed at Visual Basic or Visual C developers.

However, I can't help feeling that I should not have had to build this tool – it should have been part of my APL system! It will soon be time to re-launch APL to a new generation of developers. The people with an 'attitude' to APL (good or bad) are moving on, and I think we will soon have an opportunity to try to sell APL again. This will not be possible if the new audience feel that the mechanisms we provide for inter-application communication are significantly inferior to the environments they are used to. On the other hand – if we have it, there is no reason why we should not be able to go out and sell APL to people who need to compute something; all the other languages are still far behind APL in this respect.

If the vendors are not already working on it, this seems to be the next big challenge, which we need to address as a community. I think I would go as far as to say that the long term survival of APL depends on support for objects at the language level.

Unknown to me at the time I wrote these lines, Dyadic had already done most of the work required to instantly obsolete OOF. At APL Berlin 2000, they presented version 9, which provides improved OLE/ActiveX support. Thanks to Dyadic, this has turned into a perfect project for me: I had all the fun of learning a bit more about objects, designing it and getting it to work, but was spared the grief of packaging, rollout and support.

What's New?

A couple of new features in version 9 make the difference. Firstly, version 9 now loads all Type Information associated with an object when the first top-level object is created. This is done much faster than before and the results are saved in the workspace¹.

Secondly, you can now use the "industry dot syntax" to refer to the properties of GUI and OLE objects. So where you previously had to write:

```
'F' QWS 'Caption' 'Hello World'
```

... You can now get away with:

```
F.Caption+'Hello World'
```

When resolving a "dotted name", each segment in the name is an expression, which is evaluated in the context of the object resulting from the evaluation of the segments to the left. Unless it is the last (rightmost) expression, it must return a single namespace². This allows the use of expressions in the form:

```
XL.Workbooks.(Open 'MyBook.xls').Sheets
```

We can now avoid the explicit creation of "intermediate" objects, which in version 8 was required for every level of the hierarchy. To illustrate, let us take a look at the code example required to open a workbook and retrieve data from a range within it. In Dyalog APL version 8.x, you had to do something along the lines of:

```
'XL' QWC 'OLEClient' 'Excel.Application'
'WBS' QWC 'XL' QWG 'WorkBooks'
'BOOK' QWC 3 QNQ 'WBS' 'Open' 'MyBook.xls'
'SHEETS' QWC 'BOOK' QWG 'WorkSheets' .
'SHEET1' QWC 'SHEETS' QWG 'Item[1]' .
'RANGE' QWC 'SHEET1' QWG 'Range[A1;D3]'
ρ'RANGE' QWG 'Value'
3 4
```

¹ I'm not sure what happens if you then try to use your workspace with a different version of Excel - I assume that COM interface versioning forces a reload of the relevant information?

² The ability to return namespaces as the result of functions is also new in version 9.

Quite a mouthful! With OOF, this was reduced to:

```

#.#OOF.Add '#.XL' 'Excel.Application'
#.#OOF.Add '#.BOOK' '#.XL.Workbooks.Open' 'MyBook.xls'
ρ#.#OOF.Prop '#.BOOK.Sheets[Sheet1].Range[A1;D3].Value'
3 4

```

In version 9, all you need is:

```

'XL'[]WC'OLEClient' 'Excel.Application'
ρXL.Workbooks.(Open 'MyBook.xls').Sheets.(Item
'Sheet1').(Range 'A1:D3').Value
3 4

```

More Examples

To ensure that nobody has any illusions that OOF might still have any use (unless you are unable to upgrade to v9), here is a quick run through most of the examples in the July article.

Open a Workbook

We need this for all the subsequent examples:

```

OOF:    #.#OOF.Add '#.book' '#.XL.Workbooks.Open' 'MyBook.xls'
V9.0:   book+XL.Workbooks.Open 'MyBook.xls'

```

Invoke a Method

```

OOF:    #.#OOF.Invoke '#.XL.InchesToPoints' 1
V9.0:   XL.InchesToPoints 1

```

Get Property of an Item in a Collection

```

OOF:    (1 #.#OOF.CollProp 'Name') '#.book.Sheets'
V9.0:   book.Sheets.(Item 1).Name

```

Set Property of an Item in a Collection

```

OOF:    'Demo' (1 #.#OOF.CollProp 'Name') '#.book.Sheets'
V9.0:   book.Sheets.(Item 1).Name+'Demo'

```

Get Property from all Items in a Collection

```

OOF:    names+('*' #.#OOF.CollProp 'Name') ditto

```

```
V9.0: :With book.Sheets
      names+Count<='
      :For i :In \names ◦ (i>names)+<(Item i).Name ◦ :EndFor
      :EndWith
```

OOF seems to be doing a bit better here, but *OOF.CollProp* is a monster function containing lots of ugly code. You could write a very simple cover-function in v9, if you want to. Unfortunately, despite the fact that ...

```
3=ρsheets+book.Sheets.(Item" \Count)
```

```
and 'Demo' =sheets[1].Name
```

... Dyadic have decided to spend a bit more time pondering theoretical issues before they let us³ access the properties of an array of objects using notation like:

```
book.Sheets.(Item" \Count).Name
```

Enumerated Types

Object hierarchies often make use of Enumerated Types, in which an index or constant must belong to a declared set of named values. OOF contained a utility function to retrieve the "Index Set" of a collection:

```
#.OOF.GetIndexSet '#.book.Sheets'
1 2 3
```

And if "range" is a range within a sheet:

```
#.OOF.GetIndexSet '#.range.borders'
xlInsideHorizontal 12
xlInsideVertical   11
xlDiagonalDown     5
xlDiagonalUp       6
xlEdgeBottom       9
xlEdgeLeft         7
xlEdgeRight        10
xlEdgeTop          8
```

With OOF, you could use the names (like *xlInsideHorizontal*) or the numeric ids (12) to refer to an item in a collection.

³ I hope. Please call John Scholes or Peter Donnelly and tell them you *must* have this feature, and you want it yesterday!

In version 9:

```
3 2>range.Borders.GetPropertyInfo 'Item'
xlBordersIndex
```

If this returns 'VT_VARIANT' (as it would for `book.Sheets.Item`), the argument to `Item` is not an enumerated type and the list of valid item identifiers is simply:

```
:book.Sheets.Count
```

Otherwise, you can easily extract the type information:

```
+range.Borders.GetTypeInfo 'xlBordersIndex'
xlInsideHorizontal 12
xlInsideVertical 11
...etc...
```

It is important to note that version 9 exposes enumerated type constants as variables in the namespace corresponding to the object. So you can simply refer to:

```
range.Borders.xlInsideHorizontal
12
```

And use it as follows:

```
range.Borders.(Item xlInsideHorizontal).Color+231232
range.Borders.(Item xlInsideHorizontal).LineStyle+1
```

Or even better:

```
:With range.Borders.(Item xlInsideHorizontal)
Color LineStyle+231232 1
:EndWith
```

Conclusion

With version 9, Dyadic have come close to making it as easy for APL developers to use OLE/ActiveX components as it is for developers using Microsoft development tools. The challenge of understanding the various object models remains, but it is now much easier to re-use a code fragment found in a magazine, on the web or in documentation. As a result, APL is much more competitive with respect to Visual Basic and other mainstream development tools for Windows; our chances of selling APL to a new generation of developers is dramatically improved.

Under the covers, APL is doing exactly the same job that OOF was doing before: creating temporary objects on your behalf, in order to reach the level of the object hierarchy you are reaching for. Where OOF only deleted these "temps" when you explicitly deleted the parent object (which you had explicitly created), APL (like Visual Basic) deletes them immediately unless you assign them to a variable. This means that, although it is now very easy to reach into deep object hierarchies in fantastic one-liners, it may be wise to create temporary variables in order to prevent wholesale creation and destruction of objects, which may sometimes have a high initialisation cost. For example:

```
:For i :In \book.Sheets.(Item 1).(Range'A1:J25').Count
      (book.Sheets.(Item 1).(Range'A1:J25').Item 1).Formula+'rand()*100'4
:EndFor
```

... takes three times as long to execute as:

```
range+book.Sheets.(Item 1).(Range'A1:J25')
:With range
      :For i :In \Count
          (Item i).Formula+'=rand()*100'
      :EndFor
:EndWith
```

The above code is all a bit stupid anyway, it is a workaround for what looks like a bug in Excel. You should be able to get away with:

```
book.Sheets.(Item 1).(Range'A1:J25').Formula+25 100='=RAND()*100'
```

... which takes no time at all, but for some reason Excel treats this as if you had put the formula in quotes (as a text constant), and displays the formula without executing it. If you visit each cell in turn, click on the formula as if to edit it and hit enter, it corrects the problem⁵.

Although it seems that Microsoft's imagination may have failed in this particular case, it is becoming common for object models to contain array properties. For example, ADO database interfaces return matrices of data. Life is getting easier for the array-minded developer!

⁴ Note the parentheses starting at the left and ending after "Item i". If the parentheses were around (Item i), i would be evaluated in the cell namespace, and give a VALUE ERROR. In the second example, i is a variable inside range.

⁵ A beer at the next APL conference for anyone who has found a way to convince Excel to accept an array of formulas.

GENERAL ARTICLES

This section of Vector is intended for general readers who have a working knowledge of APL or J. Authors are encouraged to submit articles which illustrate effective APL applications.

CD Labels and More

by Cliff Reiter (*reiterc@lafayette.edu*)

Prologue

The title of this note gives away the fact that the utility described here can be useful. If I had instead used the title "Mapping Rectangles to Annular Sectors" and claimed that was exciting and useful, you would have thought that I was joking. My goal in this note is to convince you that I am not joking — not entirely.

I recently have mapped rectangular images onto a segment portion of an annulus. While I would not have described it that way at the time, this has come up in three contexts in the past couple years. The first was when some students and I took images with repetitive symmetry and created circular images that resulted from wrapping the pattern around the origin [1]. An example of that is given below. Second, as I started archiving materials on CD-R discs, I found that having an image on the CD label was handy and it was especially nice when sending a CD to other people. Wrapping text around the CD label can also be striking. Third, I created a design for printing on a flying sport disc. The design included text running around the outside of a hyperbolic pattern. I plan to say more about that in a subsequent note.

My solutions to the first two problems worked, but they were tied to the particular applications. I decided that I should think about a general utility that would work in a broad context that would include all three of these cases.

The Annular Insertion Utility

The appendix gives the definition of the conjunction `AnIns` which is our utility. Its definition is fairly straightforward; it is a couple dozen lines long. How `AnIns` works will be briefly described in the appendix. It is also defined in the script `anins.ijs` that is available for downloading from [3].

Here we will focus on the syntax and application of `AnIns`. The conjunction has two arguments that are used to define the annular region and the derived function has two arguments that are the target and insertion arrays.

Consider the following example.

```
Load 'anins'

(15 15$0) 0.5 1 AnIns 1r2p1 0 (10+i.5 8)
0 0 0 0 0 0 0 10 0 0 0 0 0 0 0
0 0 0 0 0 0 0 18 19 19 12 0 0 0 0
0 0 0 0 0 0 0 26 27 28 20 21 0 0 0
0 0 0 0 0 0 0 34 35 36 29 30 22 0 0
0 0 0 0 0 0 0 0 0 45 38 30 23 15 0
0 0 0 0 0 0 0 0 0 0 46 39 31 24 0
0 0 0 0 0 0 0 0 0 0 0 40 32 24 0
0 0 0 0 0 0 0 0 0 0 0 41 33 25 17
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

In that example, the derived function had the 15 by 15 matrix of zeros as its target array and the insertion array given by $10+i.5 \ 8$ as its right argument. Not every entry from the insertion array appeared, and some appeared more than once. That is a result of the natural distortion occurring in this type of insertion.

The left conjunction argument specifies the radial range as a fraction of the maximal inscribed radius. Thus $0.5 \ 1$ indicated that the insert was to go from one-half the maximal radius up to the maximal radius. The right argument $1r2p1 \ 0$ specifies that the angles should go from $\pi/2$ radians (90 degrees) to 0 as the insert is placed clockwise.

The argument matrices may be raw "a." data. For example, the following inserts a three-fourths radius semicircle of the letter "M" in a character array of digits.

```
(15 15$'01234') 0 0.75 AnIns 0 _1p1 (10 10$'M')
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
012340123401234
```

In general, the target and insert arrays must have the same type, but they can be higher dimensional arrays. For example, they might be arrays of RGB triples. However, the target array is assumed to be square in the sense that its first two axes have the same length—and hence the inscribed radius is the same horizontally and vertically.

The radial fractions should be between 0 and 1. A reversal of the order of those fractions results in a radial reversal of the inserted region.

The angles should be specified in radians between $-\pi$ and π . Reversal of those angles results in the reversal of the clockwise/counterclockwise sense of the insertion. Clockwise motion maintains the usual left to right order of the insertion image and hence the angles specified in our examples will usually be decreasing.

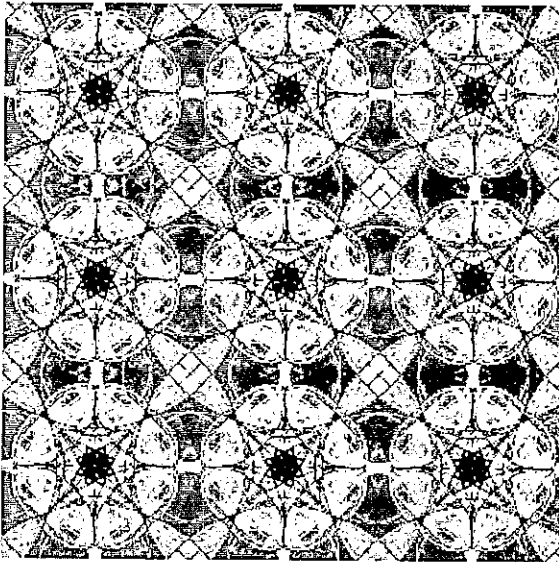


Figure 1. A Chaotic Attractor with Repetitive Symmetry

Application to Symmetric Fractals

The image *nearp4m.bmp* is shown in Figure 1. It is a chaotic attractor from [2]. We will use a function from the script *raster4.ijs* in order to read the bitmap into J. Both the image and the script are available from [3] with the materials for [2]. We assume that the image and script have been placed in a subdirectory *fj2* of the main J directory. Use a path appropriate for where you placed those files. The

result of reading the file is a boxed list giving the palette for the image and the matrix of indices into the palette. We see below that the image has 256 colors, color index 0 is white, and the image is 750 by 750.

```
load 'fvj2\raster4'  
  
'p b'=: readbmp8 'fvj2\nearp4m.bmp'  
  
$p  
256 3  
  
0{p  
255 255 255  
  
$b  
750 750
```

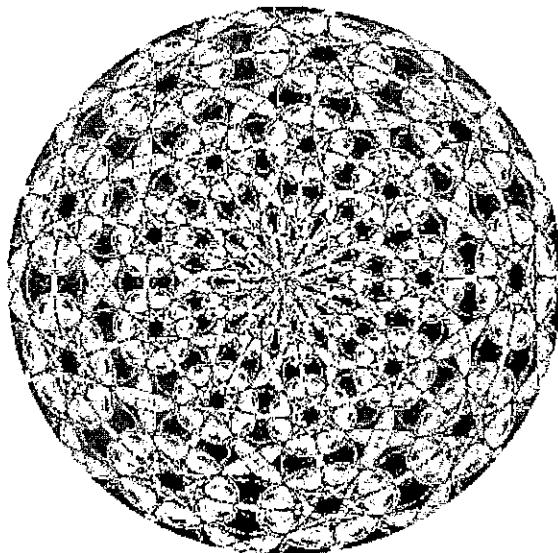


Figure 2. The Chaotic Attractor Mapped onto a Circle

Next we create a white 1000 by 1000 array and replicate the image from Figure 1 three times as our insertion array. We then apply AnIns on the full disk.

That is, the radius is from 0 to 1 and the angle from π to $-\pi$.

```
blank=: 1000 1000$0

insert=: b,.b,.b

result=: blank 0 1 AnIns 1p1 _1p1 insert

(p;result) writebmp8 'figure2.bmp'
```

The result was written as a bitmap using `writebmp8`. It appears in Figure 2.

The application of `AnIns` on a 1000 by 1000 image is not trivial. J4.04 required about 32 seconds and 10M on a 733MHz PC (J4.03 required 80 seconds and 13M—so J4.04 offers an impressive gain in speed).

Application to CD Labels

Creating CD labels using `AnIns` is easy too, although there are usually a few more details to consider. Here we will aim to create a CD label for the CD containing several dozen digital images from a trip. The image *keys.bmp* is available from [3] with materials for [2]. Assuming *raster4.ijs* is loaded and the image is in the *fvj2* directory, we can read the image as follows. Also, since this is a 24-bit image, the result is a 3-dimensional array. Thus we make our blank array a three dimensional array.

```
$insert=: readbmp24 'fvj2\keys.bmp'
300 450 3

blank=: 1000 1000 3$255
```

Since the width of a CD label is 4.5 inches and its center hole has width 1.625 inches, we should let the radius run from 0.361111 to 1 in order to lie on the CD label. We will insert the *keys.bmp* image into an angular quarter of the label.

```
1.625%4.5
0.361111

temp=: blank 0.361111 1 AnIns 1p1 1r2p1 insert

temp writebmp24 'temp.bmp'
```


We also want to put some text onto our image. Since the insertion needs to be an array representing a 24 bit image, the text needs to be rendered and saved in a suitable format. Here we find it convenient to use a utility from the script *dwin.ijs* from [2] and available at [3] to open a graphics window. Then, we place some text in the window using standard J graphics commands.

```
load 'fvj2\dwin'  
WINSHAPE=: 600 100  
dwin 'keys'  
glfont '"Times New Roman" 400 bold'  
gltextxy 10 990  
gltext 'Marilyn and Cliff    20th Anniversary'  
gltextxy 10 490  
gltext '    Trip to the Keys, December 1999'  
glshow ''  
glfile 'keystext.bmp'  
glsavebmp 1200 200
```

Now we load the file with text and insert it.

```
text=: readbmp24 'keystext.bmp'  
temp2=: temp 0.361111 1 AnIns 1r4p1 _3r4p1 text  
temp2 writebmp24 'keys_cd_label.bmp'
```

Now the file *keys_cd_label.bmp* contains our CD label. One detail remains: we need to print the label so that it is the correct size and it falls on the correct place on a sheet of blank CD labels. This may be accomplished in a Word Processor or other program that allows images to be sized and margins to be set. We print on Avery 5824 labels which requires a 4.5 inch image with a top margin of 0.5 inches and a left margin of 2 inches. We recommend a trial printing on scrap paper although the label created here printed perfectly the first time. Figure 3 shows the label on a CD.



Figure 3. A CD Label with an Image and Text

Postlogue

We have seen that the utility `AnIns` can be used to wrap images onto an annulus. This allowed us to create an interesting complex image with circular symmetry and we also used this to insert an image and text into circular sectors on a CD label.

We kept our examples brief, but consider the following possibilities. The two lines of text in Figure 3 would look better if different font sizes had been used. That can be accomplished by inserting one `glfont` command in our text construction. Also, the distortion of images is less noticeable when using smaller sectors. Using multiple images can be attractive. Our examples always began with a blank target array. How could you add text to the image from Figure 2 used as the target image? How could you create a blank circular window in the image center? What would the image look like if you repeated the image from Figure 2 and inserted it onto a CD label?

Appendix

The following is a conjunction for annular insertion. It is defined in *anins.ijs* which is available on the web at [3]. The conjunction is based on the following approach. The "while-loop" runs through each scan line of the resulting array *z*. The pixel coordinates of each scan line in the result array are converted to semi-normal coordinates by *j.&i2s*. That is, the pixel coordinates are scaled to a coordinate system with lower left *_1 _1* and upper right *1 1*. The local function *np* converts given coordinates into normalized polar coordinates. That is, radii and angles given as fractions of the positions in the region defined by the annular sector. Only results which lie between 0 and 1 in both those coordinates correspond to the insertion region and the scan line is amended to suitably change such pixels.

```

AnIns=: 2 : 0

:
imsz=#x.
'r0 r1'=.m.
'a0 a1'=.n.
i2s=.<:@(*&(2%imsz-1))      NB. image coord to seminormal
nr=.(%r0-r1)"_ * 10&o. - r1"_  NB. normalized radius
na=.(%a1-a0)"_ * 12&o. - a0"_  NB. normalized angle
np=.(nr,na)@+@j.&i2s f.      NB. normalized polar
i=.0
z=.i.0,imsz,2}.$x.
while. i<imsz do.
  npc=(i,imsz) np i
  q=.,/ |: (0&<: *. <:1)npc
  od=.i{x.
  if. 0<+/q do.
    id=.(<"1 <. 0.5+(<:2{.$y.)**1 q#npc){y.
    z=.z,id(q#i,imsz)}od
  else.
    z=.z,od
  end.
  i=.>:i
end.
z
)

```

References

- [1] Nathan C. Carter, Stephen M. Grimes, and Clifford A. Reiter, Frieze and Wallpaper Chaotic Attractors with a Polar Spin, *Computers & Graphics*, 22 6 (1998) 765-779.
- [2] Clifford A. Reiter, *Fractals, Visualization and J*, Second Edition, Jsoftware, Toronto, 2000.
- [3] Clifford A. Reiter, J materials, <http://www.lafayette.edu/~reitercf/j/index.html>.

Some More of the Personalities at APL2000

OK, Vector Production can't count we admit it so here are some more pictures from Ray's CDs



How it was, not so very long ago



... and how it is now. Three Americans, one South African, One Russian (who always obeys traffic rules), and not a weapon in sight!

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

Hacker's Corner: How big is that Doggie in the Window?	Bill Parke, Adrian Smith	108
Technical Correspondence		
Bell Numbers	Gene McDonnell	112
APL+Win Application Migration	Ajay Askoolum	
At Play with J: Blists in OLEIS	Gene McDonnell	116
The Generalized Tribonacci Sequence	Joseph De Kerf	131

Hacker's Corner - How Big is that Doggie in the Window?

original from William Parke (wrp@parkenet.org), notes by Adrian Smith

Introduction

Being kind to web browsers is always a good idea, but particularly so when it comes to images. For readers who have been asleep for the last 5 years, you include a picture in a web page with an HTML fragment like:

```
This is my dog!
```

... and when the browser hits the tag, it beetles off back to the server and asks for the image file. In fact if you have lots of pictures, sensible browsers start a whole series of threads running to get the pictures in parallel. Of course the whole of the text has usually been loaded long before the last image arrives, which leaves the browser with a big problem - how can it lay out the page until it knows how big all the images are?

There are two possible solutions here:

1. wait for the last picture to come, compute the wraparounds and gaps, then lay out the text
2. slam the text down as soon as you can and rewrite it as many times as you have to, as information about the pictures trickles in.

Option 1 is good if you have ADSL or are working on a LAN, option 2 is actually not as bad as it sounds on a slow modem. However, you can *tell* the browser exactly what to expect by including some extra information in the tag:

```
This is my dog!
```

Now the page can be rendered correctly and immediately, and the pictures will slot neatly into the gaps. Everyone deeply content, all pigs fed and ready to fly etc. But how does your APL web-site builder find out how big a picture is so it can include the information in the HTML stream that it writes out?

This was an issue I had been quietly ignoring for some time, as my HelpStuf workspace only does Webpages as a by-product, being much more focussed on Help files and manuals. However it is rapidly becoming essential to do good web-based documentation too, so I was delighted when Bill Parke took up the

challenge, took on the research and came up with a serious piece of APL+Win code which will find the size of more or less any file you care to name.

SizeofIMG in APL+Win

Bill keeps up-to-date copies of his code on his own website, and the original is a shade over 300 lines long, so it would be pointless to list it here. However the references are (very sensibly) included in the leading comments:

```

aV([fileformat width height depth hres vres units] or
[attributestring])+[returnformat] SizeofIMG filename -- find
size information in BMP, JPG, GIF and PNG image files
aVver 2000.06.09 wrp <wrp@parkenet.org>
aV file ↔ [path\]imagefilename
aV rf ↔ 0 = return raw data (default)
aV      1 = return string formatted for HTML <IMG> tag
aV z ↔ raw data: fileformat width height depth Hresolution
Vresolution resolutionunits
aV      or format string: 'width="xxx" height="yyy"'

aV JPEG information adapted from:
aV the Independent JPEG Group's JPEG software, release 6b
aV Copyright (C) 1991-1998, Thomas G. Lane.
aV specifically jdmarker.c
aV ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz
aV ftp.simtel.net/pub/simtelnet/msdos/graphics/jpegsrc6b.zip
aV
aV BMP, GIF and PNG information from:
aV Miano, John. "Compressed Image File Formats"
aV Copyright (C) 1999, ACM Press (Addison-Wesley)
aV ISBN 0201604434

```

He returns the size information, and quite a lot of other interesting stuff about the file. Calling it is as simple as it looks:

```

1 SizeOf 'd:\digipix\cars\mazda.jpg'
width=1852 height=1239

```

Here is my translation into Dyalog APL, much reduced in size as I only needed the file-size information and could ignore bitmap format as these are never used in my webpages. I have cut the leading comments, which just repeat Bill's notes.

```

v size←SizeOf file;fileformat;units;height;width;nsz;ntie;
byte;nread;length;data;⎕IO
[1] a Get size info (GIF,JPG,PNG) and return in HTML format
.....
[14] a Translated from Bill Parke's APL+Win code by ACDS - July 2000
[15] a See http://parkenet.org/wrp/apl/SizeofIMG.html for his notes

```

```

[16]
[17] size+' ' o [IO=0
[18]
[19] a Initialize data variables
[20] fileformat+units+' ' o height+width+0 o ntie+1+[/0,[NNUMS
[21]
[22] :Trap 0 a Open file with error trapping
[23]   file [NTIE ntie 0
[24] :Else
[25]   Log'Could not open ',file,' no size information will be included'
[26]   :Return
[27] :End
[28]
[29] nsize+[NSIZE ntie
[30]
[31] :If 255 216=256|[NREAD ntie,83,(nread+2),0 a is JPEG file
[32]   fileformat+'JPEG'
[33]   :While nread<nsize a read byte-by-byte, look for FF (255) markers
[34]     :Repeat a read until marker found or end of file
[35]       byte+256|[NREAD ntie,83 1
[36]       -(nsize=nread+1+nread)+END
[37]     :Until 255=byte a marker found, look for marker code
[38]
[39]     :Repeat a read marker code, skip any FF padding
[40]       byte+256|[NREAD ntie,83 1
[41]       -(nsize=nread+1+nread)+END
[42]     :Until 255=byte a length includes itself, so subtract here
[43]
[44]     length+2+256 256|256|[NREAD ntie,83 2 o nread+2+nread
[45]
[46]     :Select >byte
[47]     :CaseList (192+116)-196 200 204
[48]       data+256|[NREAD ntie,83,length
[49]       nread+nread+length
[50]       a image size is contained in next 4 bytes
[51]       height+256 256|data[1 2] o width+256 256|data[3 4]
[52]       +END
[53]
[54]     :Else a not interested -- skip this parameter segment
[55]       :If 0<length
[56]         byte+[NREAD ntie,83,length
[57]         nread+nread+length
[58]       :End
[59]
[60]     :EndSelect
[61]
[62]   :EndWhile a -- end JPEG section -----
[63]
[64] :ElseIf 'GIF'=[NREAD ntie,82 3 0
[65]   fileformat+'GIF',[NREAD ntie,82 3 a GIF subtype ('87a' or '89a')
[66]   (width height)+[NREAD ntie,163 2

```



```

[67]
[68]  a -- end GIF section -----
[69]
[70]  :ElseIf 137 80 78 71 13 10 26 10*256|[]NREAD ntie,83 8 0
[71]    length+323 []DR[]NREAD ntie,82 4
[72]    :If (13=length)^'IHDR'=[]NREAD ntie,82 4
[73]      fileformat+'PNG'
[74]      width+323 []DR[]NREAD ntie,82 4
[75]      height+323 []DR[]NREAD ntie,82 4
[76]    :End
[77]  a -- end PNG section -----
[78]
[79]  :End
[80]
[81]  END:
[82]
[83]  :If ^/0<width,height  a only supply valid values
[84]    size+' width=',(width),' height=',(height)
[85]  :End
[86]
[87]  []NUNTIE ntie

```

v

Notice the trailing 0 on the call to `[]NTIE` on line[23] – this is something I almost always forget (including the above function until I pasted it into Vector 5 minutes ago) and it is essential if you are accessing a read-only network drive. Notice the effect of correcting the problem ...

```

SizeOf '\\della\d\pix\berlin\bbgate.jpg'
Could not open '\\della\d\pix\berlin\bbgate.jpg no size
information will be included

```

```

SizeOf '\\della\d\pix\berlin\bbgate.jpg'
width=523 height=360

```

I think APL+Win defaults to opening the file read-only (option=0) but Dyalog defaults to read-write mode which is a very silly idea if you keep your pictures on a network drive, a CD ROM or a read-only ZIP drive.

Summary

If you are using APL to build web-pages, go and get Bill's code and just use it. If you really want to know about the details, follow up his excellent references. Either way, he has made a really useful contribution to the corpus of publicly available APL code, and deserves all our thanks.

TECHNICAL CORRESPONDENCE

From: Eugene McDonnell

27th September 2000

In Joseph De Kerf's "Bell Numbers and APL" in Vector 17.1 he writes, "Bell numbers are rather unknown....In most books on number theory...they are not mentioned, and this notwithstanding their practical importance." It must be something of a surprise, then, to find two articles that discuss Bell numbers in the same issue of your nonmathematical magazine: Mr. De Kerf's and mine ("Blists in OLEIS"). They are the central topic of his article, but only appear tangentially in mine. I drew on Kenneth Iverson's discussion of the Stirling subset numbers in chapter 6, page 89, of his "Concrete Math Companion". His lovely derivation uses matrix divide with powers on the left and falling factorials on the right to produce the table of Stirling subset numbers, whose row sums in turn give the Bell numbers. In his appendix Mr. De Kerf discusses this use of summing subset numbers to give the Bell numbers, and properly points out that this method, although of pedagogical interest, is quite inefficient as a way of calculating the Bell numbers. I was fascinated, as I always am when I read his articles, by the several much more efficient algorithms he showed for this purpose.

APL+Win Application Migration

From: Ajay Askoolum

31 July 2000

This response is prompted by *ZarkWin:Windows Programming without DOWI* by Gary A Bergquist and *Taking the Migraine out of Migration* by Walter G Fil which appeared in Vector Volume 17 Number 1.

My point is that successful migration is not that simple.

Which applications do the authors have in mind?

Is it APL*PLUS/PC, or APL*PLUS/PC applications already migrated into APL*PLUS/386, or applications developed in APL*PLUS/386? In other words, is the screen interface for data entry based on `INKEY` as in earlier versions of

APL*PLUS/PC or on `WIN` as in later versions of APL*PLUS/PC and APL*PLUS/386? Is the screen image based on a 24x80 character array or based on `⎕` or `⎕`?

How do these applications store the data entered on the screen?

Typically, the popular approaches were either to use a segmented string for each screen where each segment occupies one variable/slot or to use double segmented strings where each screen occupies a first level segment and its content occupied the second level or to hold each entry on all screen in one huge character array where each screen had an absolute start and end row and each entry on each screen corresponded to a particular row.

That is, either as

```
αNAMEαSEXαSALARY
```

or

```
βαNAMEαSEXαSALARY βαNINOαDATEJOINEDαDATELEFT
```

or

```
R[1;]
R[2;]
...
R[n;]
```

Therefore, each screen and the content of each screen could only be established by a hardcoded system of dead reckoning. The insertion of a new screen or a new variable in any screen or the relocation of a variable on a screen involved not only the changing of all those dead references but also a major migration task to remap the existing arrangement to the new.

And, when an application required recurring data further dead references are needed. For example, if the application could track n service period defined in terms of m variables, say DateJoined, DateLeft etc., these variables would be referred to as $(m-1)+\text{row}/\text{segment}$ 1 for DateJoined in service period 1, (in index origin 1) etc. All of this would be buried in the calculation engine of the application.

Do these applications have a menu system?

I dare say they do. There would be little point in migrating a one screen application.

Therefore, each screen would be displayed exclusively, requiring the user to go back and forth.

Each screen would also be self contained in the code that defines it: this includes lists, defaults, validation, error messages etc.

To what extent do the existing applications contain code to manage the inherent limitations of the interpreter and/or the operating system used?

Typically,

- application code always assumed the current directory or a directory referred to by a library number
- file input/output was necessitated by the size of the workspace
- printing was highly customised within APL, not least because of APL characters.

The application development cycle.

Consider the application development cycle. It is *Program, Test, Debug, Document* in the first phase and *Maintain, and Evolve* in the second phase. Migration is subject to the same development cycle except that it necessitates another cycle, namely, *Integrate*.

It is well established that phase 2 is the most expensive in the application life cycle.

The articles seem to suggest that phase 1 is all that is required and all other things are constants. Not so. At least not so without serious loss of credibility.

An application using the features of APL+Win version 3.5 makes a comparable one written in version 3.0 look dated. What of an application that looks and feels like a DOS APL application?

The migration route suggested still incurs all the overhead of phase 1 with the possible exception of the *Document* cycle (typical APL approach) but fails to realise any of the potential benefits and does not consider phase 2 at all.

Consider the missed opportunities:

Moving the screen input variables outside the application, say to a spreadsheet where all the properties/characteristics are defined, thereby allowing easy

maintenance (in terms of tracking duplicate names, ensuring consistent validation, error reporting etc) of the 'Input' segment of the application.

Usage of the selector interface, housing each screen on a separate tab thereby allowing open access to any screen at the click of a mouse.

Holding the values entered in named variables which retain their inherent type, such as numeric, date or character.

Usage of third party ActiveX controls such grid objects to provide a intuitive interface to the user, particularly for recurring items such as the one for service period described above. Usage of nested arrays is an inherent part of this.

Usage of the operating system in two critical ways, namely resources and API calls. For instance, using the Windows printing facility allows the selection of printers on a network (including their particular facilities such as duplex printing) can be available without any programming overhead other than the call. The call to the CharUpper API is far more efficient than any equivalent in APL.

Usage of other resources on the computer, such as COM objects for sending/receiving data to/from standard applications. Something like WORD and EXCEL print and present data far better than APL.

Usage of control structures to make program flow more visible and self documenting.

Usage of a more modern idiom library such as IBM's is far more programmer efficient and powerful than, say, the original FINN APL idiom library. Refer to the algorithms for formatting without leading or trailing blanks, for instance.

Conclusion

In my view, a legacy system is neither the specification nor the boundary of a Windows application, it is simply the test bed.

The key to a successful application is *integration* with the platform it runs on and with the applications it co-exists with. The efficiency of that application is in using the features of the tool it is written with, and not in maintaining the quirks of previous incarnations of that tool.

At Play with J: Someone Just Moved! Who Was It? or, Apter's Puzzle

by Eugene McDonnell

The Puzzle

Stevan Apter recently proposed this puzzle to the online K group: Given a list of distinct items, and a second list changed by moving only one item of the original, find which item has been moved. There was a fair amount of discussion about this, and a number of proposed solutions, a surprising number of which were erroneous. This paper gives a solution that I believe works properly in all cases. The greater part of the paper, however, discusses the reverse problem: Given a solution, find the list that gave rise to it.

Rotated Infix Permutations

First, to solve Apter's problem: Given the two lists A and B:

```

A
3 1 4 5 9 2 6
B
3 1 5 9 2 4 6

```

tell which item in A was moved in order to produce B. It couldn't have been the 5 or 9 or 2 because they form an infix in the order of the original. The 4, which had preceded 5 and 9 and 2 is the one that is out of order: it is now at their right: it has been moved from index 2 to index 5.

The problem is simplified if the items are replaced by their indices. Since the items are distinct, A can be replaced by the identity permutation, and B by the indices of its items in A. For example:

```

C =: i. # A
D =: A i. B
C ,: D
0 1 2 3 4 5 6
0 1 3 4 5 2 6

```

C and D contain all the information needed to solve the original A and B problem. In fact, D is all that is needed: the identity permutation C can be understood. In

what follows, I'll assume that a problem list is given in this D form. Thus an argument to the solution program might be:

$$\begin{array}{c} D \\ 0 \ 1 \ 3 \ 4 \ 5 \ 2 \ 6 \end{array}$$

Comparing D to C shows that some of the items remain in their original positions, but others have been moved. The following shows the items that have been moved in bold. These form a rotated infix; because of this I'll call D a *rotated infix permutation*, which for convenience will be abbreviated to *rip*.

$$0 \ 1 \ 3 \ 4 \ 5 \ 2 \ 6$$

The nonzero items of D-C show which have been moved:

$$\begin{array}{c} D-C \\ 0 \ 0 \ 1 \ 1 \ 1 \ _3 \ 0 \end{array}$$

Those which have been displaced by the move produce a 1 in the difference; the one moved produces 3. But an item can be moved to a lower position as well as to a higher. Suppose we are given to solve:

$$\begin{array}{c} E \\ 0 \ 1 \ 5 \ 2 \ 3 \ 4 \ 6 \end{array}$$

The items in bold again represent a 1-rotation, but this time it is to the right. If we subtract C from this we get:

$$\begin{array}{c} E-C \\ 0 \ 0 \ 3 \ _1 \ _1 \ _1 \ 0 \end{array}$$

Here the value corresponding to the moved item is positive value, and the displaced items produce 1s. Considering both cases, it is evident that the moved item is determined by finding the maximum magnitude in the difference between the list to solve and the identity permutation. There are two difficulties to discuss. The first difficulty: suppose we move an item just one position to the right or left: what results?

$$\begin{array}{c} F = .0 \ 1 \ 3 \ 2 \ 4 \ 5 \ 6 \\ F-C \\ 0 \ 0 \ 1 \ _1 \ 0 \ 0 \ 0 \end{array}$$

The magnitudes of F-C show two possible results. It's unclear whether the 3 has been moved to position 2, or the 2 has been moved to position 3. Either one can be chosen. The rule used here is: Among equal maxima, choose the one occurring

first. This is easiest because of the way *J's Index of primitive* (*i.*) is defined, because it gives the index of the first occurrence of the item sought. This primitive has right of seniority over *J's Index of Last primitive* (*i:*), which is a relative newcomer; Index of antedates even APL: it is described in Iverson's 1962 book *A Programming Language* (Wiley, 1962) in section 1.16 *Ranking*, page 31:

The *rank* or *index* of an element $c \in b$ is called the *b index of c* and is defined as the smallest value of i such that $c = b_i$.

Thus, for the rip F above, 3 will be identified as the item that has been moved – even though it might in fact have been produced by moving 2 to position 3.

The second difficulty arises from the possibility, not excluded in Apter's statement of the problem, of moving an item from its original position to its original position. For example, the list:

```

      G
    0 1 2 3 4 5 6
  
```

if proposed as a problem to solve, might be the result of moving any of the seven items back to its original position. What we get if we subtract C from G is:

```

      G - C
    0 0 0 0 0 0 0
  
```

Again following the rule *among equal maxima, choose the first* I would find 0 as the one having been moved (to 0).

Canonical Specifications

A rip such as D, E, F, or G can be represented by a list of three integers: its length L, the initial position I of the moved item, and its final position F. For example, the rip D is specified by 7 2 5; E by 7 5 2; F by 7 3 3; and G by 7 0 0. I'll call such a list the rips' *canonical representation*, or *casp*.

The function CfR below solves Apter's problem, yielding the casp from the rip:

```

CfR =: monad define"1
NB. casp from rip
R=.y.
F=. (i.>.) | (-i.@#)R
L=.#R
I=.F{R
L,I,F
)
  
```


Applying it to the sample rips:

```

    Cfr D
  7 2 5
    Cfr E
  7 5 2
    Cfr F
  7 3 2
    Cfr G
  7 0 0

```

For aficionados of the one liner:

```
OLCfr = . # , (([ , ~ {) ~ (i. >. /) @ ([: | ] - [: i. #)) ~
```

A function inverse to Cfr would take a casp L,I,F and yield the rip which gave rise to it. It can be described informally like this:

lay out a row of L numbered blocks

0	1	2	3	4	5	6
---	---	---	---	---	---	---

remove the Ith block (which has the number I on it) and set it aside

0	1	3	4	5	6
---	---	---	---	---	---

2

collapse the remaining blocks over the empty space

0	1	3	4	5	6
---	---	---	---	---	---

separate these into two parts, the first F and the rest

0	1	3	4	5	6
---	---	---	---	---	---

and insert the removed block in the space made (which is F)

0	1	3	4	5	2	6
---	---	---	---	---	---	---

This function RfC which, given a casp, yields a rip, is:

```
RfC =: monad define"1
'L I F'=.y.
M=.I-.~i.L
(F{.M),I,(F).M
)
```

Here are some uses of RfC:

```
RfC 7 2 5
0 1 3 4 5 2 6
RfC 7 5 2
0 1 5 2 3 4 6
RfC 7 3 2
0 1 3 2 4 5 6
RfC 7 0 0
0 1 2 3 4 5 6
```

Roughly speaking, CfR and RfC are inverses, and thus we'd like to be able to say that:

```
P -: RfC CfR P
```

and

```
C -: CfR RfC C
```

The function MC below, given a positive integer argument, yields a 3-column table of all the casps for rips of that length. For a list of length n, there are n^2 rips, one for each initial position versus each final position. A variation of the odometer function is required. Given a length L, form i. *: L, then the (L,L) representation of each, and finally, prefix L to each, ending with an L² by 3 table.:

```
MC=:13 : 'y... (2#y.)#:i.*:y.'
]C3=.MC 3
3 0 0
3 0 1
3 0 2
3 1 0
3 1 1
3 1 2
3 2 0
3 2 1
3 2 2
```

Next, I'll get the corresponding rips:

```
R3 =. RfC C3
```

And finally, put C3 next to R3 so we can see the correspondences:

```

  2 2 2 6 2 2": C3, .R3
3 0 0      0 1 2
3 0 1      1 0 2
3 0 2      1 2 0
3 1 0      1 0 2
3 1 1      0 1 2
3 1 2      0 2 1
3 2 0      2 0 1
3 2 1      0 2 1
3 2 2      0 1 2

```

The number of distinct results is not L^2 , but less than that. When I and F are the same the results are the same: each of 3 0 0, 3 1 1, and 3 2 2 yields 0 1 2. From these three solutions we get only one casp, so we can reduce the initial 9 by 2. In general, the diminishment coming from this case is $L-1$. Next, the results are the same when I and F are adjacent numbers, as in 3 0 1 and 3 1 0, and 3 1 2 and 3 2 1. From these four casps we get only two rips, so we can reduce the total by another 2. In general, the diminishment coming from this case is also $L-1$, one for each adjacent pair. The 9 results are thus reduced by another 2, leaving just 5. The general formula for the number of distinct rips is $L^2-2(L-1)$, or, simplified, L^2-2L+2 , which gives the J polynomial function `2_2 1&p.`

The function NR below, given an integer argument, yields the number of distinct rips of that length:

```

NR=.2_2 1&p.
(, .NR)>.i.10
1 1
2 2
3 5
4 10
5 17
6 26
7 37
8 50
9 65
10 82

```

The Anatomy of Rips

I've been in the habit of checking sequences such as the second column above against the entries in Neil Sloane's invaluable collection, which appeared in print first in his *A Handbook of Integer Sequences* (Academic Press, 1973). Recently the book has been supplemented by an ever growing collection on his web site, *On-Line Encyclopedia of Integer Sequences*, at:

<http://www.research.att.com/~njas/sequences/>

The series 1 2 5 10... is ID Number A002522, which describes it in offset 0 as n^2+1 . My series is offset 1, which implies the formula previously given, namely n^2-2n+2 . It also notes that this sequence is the "Left edge of A055096" and if you refer to this sequence you will find that it is a triangle like Pascal's, where the entries are sums of distinct squares:

				5				
		10		13				
	17		20		25			
	26	29	34		41			
37	40	45	52		61			

The left edge is a truncated version of our series, lacking its first two items. Pursuing this any further will take me on too wide a detour from my main goal: the analysis of rips; but I shall be referring often to Sloane's Encyclopedia in what follows.

I began my study of rips by finding the *anagram index* of a fair number of them, using J's A. primitive, and listing on a sheet of paper those of length seven in order by length of rotated infix, and within that by highest maximum I and F. Here are those with rotated infix of length 3, and the associated anagram index (I've put the disordered infix items in bold type).

I<F									
L I F			rip					A.	
7 4 6	0	1	2 3 5 6 4					3	
7 3 5	0	1	2 4 5 3 6					8	
7 2 4	0	1	3 4 2 5 6					30	
7 1 3	0	2	3 1 4 5 6					144	
7 0 2	1	2	0 3 4 5 6					840	

I>F										
L	I	F	rip							A.
7	6	4	0	1	2	3	6	4	5	4
7	5	3	0	1	2	5	3	4	6	12
7	4	2	0	1	4	2	3	5	6	48
7	3	1	0	3	1	2	4	5	6	240
7	2	0	2	0	1	3	4	5	6	1440

The obvious way of entering the data thus gathered was to identify the rows with I and the columns with F, and to put the anagram index in item for I,F in row I, column F. This was unsatisfactory, since the numbers were going in what seemed the wrong direction, so instead I made tables where the rows were numbered by how far the right edge of the infix was from the right end of the table, and the columns were numbered by the length of the infix. Thus the anagram index of 30 where the L I F is 7 2 4, which has the infix 2 spaces from the right edge and has length 3 would be entered at row 2 column 3 of the I<F table.

		I<F						
		1	2	3	4	5	6	7
0	0	1	3	9	33	153	873	
1	0	2	8	32	152	872		
2	0	6	30	150	870			
3	0	24	144	864				
4	0	120	840					
5	0	720						
6	0							

I>F								
		1	2	3	4	5	6	7
0	0	1	4	18	96	600	4320	
1	0	2	12	72	480	3600		
2	0	6	48	360	2880			
3	0	24	240	2160				
4	0	120	1440					
5	0	720						
6	0							

Studying these tables convinced me that I could see the rule determining the value of an entry. For the I<F table, column 1 would always be zero; an infix of length one meant that I and F were the same, so the result would be the identity permutation, which is permutation 0 for permutations of all lengths. Column 2 would always be $!(1+\text{row number})$. Column j for $j>2$ would be sums of $(j-1)$ successive items of column 2. For example, the entries in column 3 would be $1+2$, $2+6$, $6+24$,...; in column 4 would be $1+2+6$, $2+6+24$, $6+24+120$,... and so forth. This can be shown using J's Infix adverb, $x\ u\ y$, where the function u (in our case $+/$) is applied over successive length- x infixes of y :

```

]fs=.!i.10
1 1 2 6 24 120 720 5040 40320 362880
  2+/\fs
2 3 8 30 144 840 5760 45360 403200
  3+/\fs
4 9 32 150 864 5880 46080 408240
  4+/\fs
10 33 152 870 5904 46200 408960
    
```

For the I>F table, the entries in column 1 and 2 would be the same as in the I<F table, for the same reasons. Column j for j>2 would be (j-1)*(j-2) drop column 2. For example, the entries in column 3 would be 2*1 drop 1 2 6 24 120...; in column 4 would be 3*2 drop 1 2 6 24 120;... and so forth.

I was able to verify my conjectures for both tables after a few experiments, and wrote two functions that would give me the value for any entry in either table:

```

ILF =: 13 : '+/!(x.+1)+i.y.-1'"0
IGF =: 13 : '(y.-1)*!(y.-1)+x.'"0
    
```

```

3 ILF 4
864
3 IGF 4
2160
    
```

The *by*, *over*, and *tab* functions below come from J's Help | Phrases | 2.c

```

dl6=: by=: ' 's;@,.@[,.]
dl7=: over=: ({.;;.)@":@,
al8=: tab=: 1 : '[ by ]over x./'
    
```

(i.7) ILF tab 1+i.7x

	1	2	3	4	5	6	7
0	0	1	3	9	33	153	873
1	0	2	8	32	152	872	5912
2	0	6	30	150	870	5910	46230
3	0	24	144	864	5904	46224	409104
4	0	120	840	5880	46200	409080	4037880
5	0	720	5760	46080	408960	4037760	43954560
6	0	5040	45360	408240	4037040	43953840	522955440

(i.7) IGF tab 1+i.7x

	1	2	3	4	5	6	7
0	0	1	4	18	96	600	4320
1	0	2	12	72	480	3600	30240
2	0	6	48	360	2880	25200	241920
3	0	24	240	2160	20160	201600	2177280
4	0	120	1440	15120	161280	1814400	21772800
5	0	720	10080	120960	1451520	18144000	239500800
6	0	5040	80640	1088640	14515200	199584000	2874009600

In both ILF and IGF the arguments are modified by adding or subtracting 1. I wondered whether I could get any further insights by writing versions of ILF and IGF in which the arguments were not offset.

```
ILFx=:13 : '+!/x.+i.y.'"0
ILFx tab~i.7x
```

	0	1	2	3	4	5	6	
0	0	1	2	4	10	34	154	3422
1	0	1	3	9	33	153	873	7489
2	0	2	8	32	152	872	5912	54116
3	0	6	30	150	870	5910	46230	54117
4	0	24	144	864	5904	46224	409104	54118
5	0	120	840	5880	46200	409080	4037880	
6	0	720	5760	46080	408960	4037760	43954560	

4 142 1048

```
IGFx=:13 : 'y.*!y.+x.'"0
IGFx tab~i.7x
```

	0	1	2	3	4	5	6	
0	0	1	4	18	96	600	4320	1563
1	0	2	12	72	480	3600	30240	18931
2	0	6	48	360	2880	25200	241920	52571
3	0	24	240	2160	20160	201600	2177280	52520
4	0	120	1440	15120	161280	1814400	21772800	52557
5	0	720	10080	120960	1451520	18144000	239500800	52521
6	0	5040	80640	1088640	14515200	199584000	2874009600	

4 142 52849 52560 52578 52648

I've put numbers to the right of those rows and at the bottom of those columns which correspond to entries in Sloane's Encyclopedia. The encyclopedia doesn't contain entries for all the rows and columns of ILFx and IGFx, but they are easily obtained.

The functions below give the first ten items of the indicated row or column:

```
NB. x items of row y of ILFx
  ILFI=.13 : '+/\!y.+i.x:x.'
NB. x items of column y of ILFx
  ILFJ=.13 : 'y.+/\!i.<:y.+x:x.'
NB. x items of row y of IGFx
  IGFI=.13 : '(!y.+i.10x)*i.x:x.'
NB. x items of column y of IGFx
  IGFJ=.13 : 'y.*!y.+i.x:x.'
```

And here they are, applied to 10 items of row 3 and column 3 for each table:

```
10 ILFI 3
6 30 150 870 5910 46230 409110 4037910 43954710 522956310
10 ILFJ 3
4 9 32 150 864 5880 46080 408240 4032000 43908480
10 IGFI 3
0 24 240 2160 20160 201600 2177280 25401600 319334400 4311014400
10 IGFJ 3
18 72 360 2160 15120 120960 1088640 10886400 119750400 1437004800
```

Sloane contains tables and triangles as well as linear sequences. Table ILFx is closely related to Sloane's sequence 54115:

```
      1
     1 1
    1 2 3
   1 6 8 9
  1 24 30 32 33
 1 120 144 150 152 151
```

The formula for this triangular array T is given as:

Triangular array generated by its row sums: $T(n,0)=1$ for $n \geq 1$,
 $T(n,1)=r(n-1)$, $T(n,k)=T(n,k-1)+r(n-k)$ for $k=2,3,\dots,n$, $n \geq 2$, $r(h)=$ sum of
the numbers in row h of T.

The rows of this triangle are derived from ILFx by reading the counterdiagonals from left to right. We can get the counterdiagonals in J by using J's Box and Oblique:

..7{.</.j:ILFj/~1.7

0
0 1
0 1 2
0 2 3 4
0 6 8 9 10
0 24 30 32 33 34
0 120 144 150 152 153 154

Here it is in triangular form:

```

      0
     0 1
    0 1 2
   0 2 3 4
  0 6 8 9 10
 0 24 30 32 33 34
0 120 144 150 152 153 154
    
```

Table IGFj is closely related to Sloane's 51683:

```

          1
         2  4
        6 12 18
       24 48 72 96
      120 240 360 480 600
     720 1440 2160 2880 3600 4320
    
```

The rule for this triangle is given in 1-offset as Table: $a(n, k) = n! * k$; they are just integer multiples of factorials. For example, $a(5,3) = 5! * 3$, or $120 * 3$ or 360 .

The J *Phrases* book gives only a limited number of functions concerning rips.

NB. Phrases from 7.A

NB. Rotate last three to the left

m7 =: 3&A.

NB. Rotate last three right

m8 =: 4&A.

NB. Rotate last x to the left

```
d9 =: ([: +/[[:![:].[:i.[] A. ]
```

NB. Rotate last x to the right

```
d10=: (!@[ - !@<:@[] A. ]
```

The functions given here expand the possibilities significantly. They are of dubious practical value, because the factorials grow so large so quickly that they really aren't practical to generate rips of any great length. For that, the function RfC is far more practical.

Contracurrency

All the while I was working on this material it had been bothering me that the Anagram index grew with respect to the end of the rip, not the beginning. For example,

```

A. 1 2 0
3
A. 0 2 3 1
3
A. 0 1 3 4 2
3
A. 0 1 2 4 5 3
3
```

The Anagram index is the same regardless of the length of the rip, when the infix is the same distance from the *right* end. When the infix is the same distance from the front end, it varies:

```

A. 1 2 0
3
A. 1 2 0 3
8
A. 1 2 0 3 4
30
A. 1 2 0 3 4 5
144
```

J supports contracurrent indexing, which is right-end oriented: the last item has contracurrent index 1, the penultimate has 2, and the antepenultimate has 3, regardless of the number of items. The previous tables were labelled by length of infix and distance from right edge, with the direction of 1-rotation used to

distinguish two separate tables, as produced by the functions ILF and IGF. The function below produces a table labelled by contracurrent indices:

```
RfMC=: 13 : '{-y.})\A.RfC MC y.'
```

This forms the table of all casps for rips of length y., then obtains the rips, next obtains the Anagram indices, and last, reshapes this into a square table.

```
] q =. |.->:i.8
-8 7 6 5 4 3 2 1
  q by q over RfMC 8
```

	_8	_7	_6	_5	_4	_3	_2	_1
_8	0	5040	5760	5880	5904	5910	5912	5913
_7	5040	0	720	840	864	870	872	873
_6	10080	720	0	120	144	150	152	153
_5	15120	1440	120	0	24	30	32	33
_4	20160	2160	240	24	0	6	8	9
_3	25200	2880	360	48	6	0	2	3
_2	30240	3600	480	72	12	2	0	1
_1	35280	4320	600	96	18	4	1	0

Entry (i,j) in this table is the Anagram index of a rip of any length in which item i has been moved to index j, where i and j are contracurrent indices.

Here are functions to convert between direct casps and contracurrent casps (ccasps):

NB. contracurrent casp from direct

```
CCfD =: --` , `:3"1
```

NB. direct casp from contracurrent

```
DfCC =: (|@<. + 0: , , )/"1
```

Here is a little experiment with the above two functions which shows the limitations of the ccasp form:

```

MC3 =. MC 3
MCC3 =. CCfD MC3
MC3x =. DfCC MCC3
MCC3x =. CCfD MCC3x
format =. 2 2 2 5 3 5 2 2 5 3
format ": MC3, .MCC3, .MC3x, .MCC3x
3 0 0  _3  _3  3 0 0  _3  _3
3 0 1  _3  _2  3 0 1  _3  _2
3 0 2  _3  _1  3 0 2  _3  _1
3 1 0  _2  _3  3 1 0  _2  _3
3 1 1  _2  _2  2 0 0  _2  _2
3 1 2  _2  _1  2 0 1  _2  _1
3 2 0  _1  _3  3 2 0  _1  _3
3 2 1  _1  _2  2 1 0  _1  _2
3 2 2  _1  _1  1 0 0  _1  _1

```

A ccasp needs only two items: the contracurrent indices of the item moved and where it is moved to. This loses the information of the length of the rip involved, and so the conversion from ccasp back to casp is not exact: the length imputed is the magnitude of the minimum item. For example, 3 2 1 is converted to _1 _2, but the back conversion is 2 1 0, since the magnitude of the minimum item _2 is 2. However, 2 1 0 is converted exactly back to _1 _2.

The Generalized Tribonacci Sequence

by Joseph De Kerf

Abstract

In a previous paper we treated the Horadam Sequences, the sequences based on the second order linear recursion algorithm. In this paper we treat the Tribonacci Sequences, those based on the third order linear recursion algorithm. APL defined functions for generating these sequences are given. Benchmarks demonstrate that, just as for the Horadam Sequences, explicit calculation from the auxiliary equation is much more efficient in CPU time than the calculation based on recursion.

Introduction

In a previous paper (1) we treated the Generalized Fibonacci Sequences or Horadam Sequences (2), the sequences based on the second order linear recursion algorithm:

$$\begin{aligned} H_1 &= p \\ H_2 &= q \\ H_n &= rH_{n-2} + sH_{n-1} \quad (\text{for } n > 2) \end{aligned}$$

It was shown how explicit calculation of these sequences from the auxiliary equation is much more efficient in cpu time than generating the sequences from the recursion algorithm. In this paper we treat the Generalized Tribonacci Sequences, the sequences generated by the third order linear recursion algorithm.

Tribonacci Sequences

Originally, the Tribonacci Sequence was defined as the sequence generated by the third order linear recursion algorithm (3)(4):

$$\begin{aligned} T_1 &= 1 \\ T_2 &= 1 \\ T_3 &= 2 \\ T_n &= T_{n-3} + T_{n-2} + T_{n-1} \quad (\text{for } n > 3) \end{aligned}$$

such that $T_n = 1 \ 1 \ 2 \ 4 \ 7 \ 13 \ 24 \ 44 \ 81 \ 149 \dots$

It may be shown that the sums of the ascending diagonals of the Wong and Maddockx triangle form the Tribonacci Sequence (5)(6).

It may look strange that T_3 is set to 2 instead of 1, but the underlying idea is the same as for the Fibonacci Sequence. The Fibonacci Sequence starts with the initial values $F_1 = 1$ and $F_2 = 1$. If we set $F_0 = 0$ and $F_1 = 1$, we get $F_2 = F_0 + F_1 = 1$. In the same way, if we set $T_0 = 0$ and $T_1 = T_2 = 1$ we get $T_3 = T_0 + T_1 + T_2 = 0 + 1 + 1 = 2$.

In most current literature (7) (8), the recursion algorithm as defined above is generalized to the form:

$$\begin{aligned} T_1 &= f \\ T_2 &= g \\ T_3 &= h \\ T_n &= aT_{n-3} + bT_{n-2} + cT_{n-1} \quad (\text{for } n > 3) \end{aligned}$$

where the parameters $f g h$ and $a b c$ are any reals. We assume however $a \neq 0$, as for $a = 0$ the sequence T_n becomes independent of the initial value $T_1 = f$.

The question arises whether, such as for the Horadam Sequences, direct calculation of the Generalized Tribonacci Sequences from their auxiliary equation is much more efficient in cpu time than generating the sequences from the recursion algorithm.

The Auxiliary Equation

The auxiliary or characteristic equation associated with the third order linear recursion algorithm given above is the third degree or cubic equation:

$$t^3 - ct^2 - bt - a = 0$$

By linear transformation, this equation may be reduced to the form whose coefficient of the second degree term is zero:

$$t = s + \frac{c}{3}$$

such that $s^3 + 3ps + 2q = 0$

with $p = -(3b + c^2) / 9$

and $q = -(27a + 9bc + 2c^3) / 54$

The algorithm for calculating the roots of the equation depends on the signum of the discriminant Δ :

$$\Delta = p^3 + q^2$$

1. If $\Delta < 0$, the equation has three different real roots:

$$\begin{aligned} s_1 &= 2 \cdot r \cdot \cos \frac{\varphi}{3} && \text{with } r = \sqrt{-p} \\ s_2 &= 2 \cdot r \cdot \cos \left(\frac{\varphi}{3} - 120^\circ \right) && \text{and } \varphi = \arccos \left(\frac{q}{pr} \right) \\ s_3 &= 2 \cdot r \cdot \cos \left(\frac{\varphi}{3} + 120^\circ \right) \end{aligned}$$

2. If $\Delta = 0$ and $pq \neq 0$, the equation has a simple and a double root:

$$s_1 = -2 \cdot q^{1/3} \quad \text{and} \quad s_2 = s_3 = q^{1/3}$$

3. If $\Delta = 0$ and $pq = 0$, the equation has the triple root 0:

$$s_1 = s_2 = s_3 = 0$$

4. If $\Delta > 0$, the equation has a real root and two conjugate complex roots:

$$\begin{aligned} s_1 &= u + v && s_2 = s_r + i \cdot s_i && s_3 = s_r - i \cdot s_i \\ & && s_r = -(u + v) / 2 && s_i = \sqrt{3}(|u - v|) / 2 \end{aligned}$$

$$\text{with } u = (-q + \sqrt{\Delta})^{1/3} \quad \text{and } v = -(q + \sqrt{\Delta})^{1/3}$$

In fact, the case $\Delta = 0$ may be treated by the same formula as for $\Delta < 0$ and by the same formula as for $\Delta > 0$. Treating the case separately however, is much simpler and more accurate. Finally it should be noticed that, as we assumed $a \neq 0$ and $a = t_1 x t_2 x t_3$, the roots t are always different from 0.

More details about the solution of the third degree or cubic equation may be found in (9). An APL defined function *CUBIC*, based on the algorithm described above but treating the case $\Delta = 0$ with the same procedure as for $\Delta > 0$ has been published in (10). This means that if $\Delta = 0$, the value of s_i may be slightly different from 0, and the function *CUBIC* returns two conjugate complex roots instead of the double or triple root.

Explicit Calculation of T_n

Depending on the signum of the discriminant Δ of the s-form of the auxiliary or characteristic equation, we have:

1. If $\Delta < 0$: $T_n = xt_1^n + yt_2^n + zt_3^n$ with

$$xt_1 + yt_2 + zt_3 = f$$

$$xt_1^2 + yt_2^2 + zt_3^2 = g$$

$$xt_1^3 + yt_2^3 + zt_3^3 = h$$

2. If $\Delta = 0$ and $pq \neq 0$: $T_n = xt_1^n + (y+nz)t_2^n$ with

$$xt_1 + (y+z)t_2 = f \quad \text{or} \quad xt_1 + yt_2 + zt_2 = f$$

$$xt_1^2 + (y+2z)t_2^2 = g \quad \quad \quad xt_1^2 + yt_2^2 + 2zt_2^2 = g$$

$$xt_1^3 + (y+3z)t_2^3 = h \quad \quad \quad xt_1^3 + yt_2^3 + 3zt_2^3 = h$$

3. If $\Delta = 0$ and $pq = 0$: $T_n = (x+ny+n^2z)t_1^n$ with

$$(x+y+z)t_1 = f \quad \text{or} \quad x+y+z = f/t_1$$

$$(x+2y+4z)t_1^2 = g \quad \quad \quad x+2y+4z = g/t_1^2$$

$$(x+3y+9z)t_1^3 = h \quad \quad \quad x+3y+9z = h/t_1^3$$

4. If $\Delta > 0$: $T_n = xt_1^n + (y \cos n\theta + z \sin n\theta)\rho^n$ with

$$xt_1 + (y \cos \theta + z \sin \theta)\rho = f \quad \text{or} \quad xt_1 + y\rho \cos \theta + z\rho \sin \theta = f$$

$$xt_1^2 + (y \cos 2\theta + z \sin 2\theta)\rho^2 = g \quad \quad \quad xt_1^2 + y\rho^2 \cos 2\theta + z\rho^2 \sin 2\theta = g$$

$$xt_1^3 + (y \cos 3\theta + z \sin 3\theta)\rho^3 = h \quad \quad \quad xt_1^3 + y\rho^3 \cos 3\theta + z\rho^3 \sin 3\theta = h$$

and $\rho = \pm \sqrt{t_r^2 + t_i^2}$ (+ for $t_r \geq 0$ and - for $t_r < 0$)

$$\theta = \arcsin(t_i/\rho)$$

Defined Functions

APL defined functions for calculating the sequence $T_1 \dots T_n$, based on implicit recursion (*TRIB1*), programming of a recursive loop (*TRIB2*), and explicit calculation from the auxiliary or characteristic equation (*TRIB3*) are given below:


```

v Z+L TRIB1 N
[1] +(N<=3)/0,Z+(N[3]ρL
[2] Z+L TRIB1 N-1
[3] Z+Z,+/(3+L)×-3+Z

```

v

```

v Z+L TRIB2 N
[1] +(N<=3)/0,Z+(N[3]ρL
[2] END:+(N>ρZ+Z,+/(3+L)×-3+Z)/END

```

v

```

v Z+L TRIB3 N;P;Q;Δ;PHI;T;XYZ;UV;TR;TI;RHO;TETA
[1] +(N<=3)/0,Z+(N[3]ρL
[2] P←-((3×L[5])+L[6]*2)+9
[3] Q←-((27×L[4])+L[6]×(9×L[5])+2×L[6]*2)+54
[4] →(0 1=×Δ←(Q*2)+P*3)/ΔZ,ΔP
[5] PHI←-2oQ±P×R+(-P)×0.5
[6] T←(L[6]÷3)+2×R×2o(PHI+0,-1 1×o2)÷3
[7] XYZ←(3+L)⊕qT°. *i3
[8] XYZ←(1E-15<|XYZ)×XYZ
[9] →0,Z+Z,(XYZ[1]×T[1]*N)+(XYZ[2]×T[2]*N)+XYZ[3]×T[3]*N+3+iN
[10] ΔZ:→(Q=0)/QZ
[11] T←(L[6]÷3)+(-2 1 1)×Q*+3
[12] XYZ←(3+L)⊕((i3)°. *o 0 1)×qT°. *i3
[13] XYZ←(1E-15<|XYZ)×XYZ
[14] →0,Z+Z,(XYZ[1]×T[1]*N)+(XYZ[2]+N×XYZ[3])×T[2]*N+3+iN
[15] QZ:T←L[6]÷3
[16] XYZ←((3+L)+T* i3)⊕(i3)°. *o 1 2
[17] XYZ←(1E-15<|XYZ)×XYZ
[18] →0,Z+Z,(XYZ[1]+N×XYZ[2]+N×XYZ[3])×T*N+3+iN
[19] ΔP:UV←(×UV)×(|UV←-Q+(-1 1)×Δ*0.5)*÷3
[20] T←(L[6]÷3)++/UV
[21] TR←(L[6]÷3)-0.5×+/UV
[22] TI←(0.75*0.5)×|-/UV
[23] RHO←(TR+|TR)×((TR*2)+TI*2)*0.5
[24] TETA←-1oTI+RHO
[25] XYZ←(3+L)⊕q(T* i3).[1](2 1°.oTETA×i3)×2 3ρRHO* i3
[26] XYZ←(1E-15<|XYZ)×XYZ
[27] Z+Z,(XYZ[1]×T*N)+((XYZ[2]×2oTETA*N)+XYZ[3]×1oTETA*N)×RHO*N+3+iN

```

v

```

L+1 1 2 1 1 1
L TRIB1 10
1 1 2 4 7 13 24 44 81 149
L TRIB2 10
1 1 2 4 7 13 24 44 81 149
L TRIB3 10
1 1 2 4 7 13 24 44 81 149

```

The left argument L is a six-element vector containing the initial values $f g h$ and the parameters $a b c$. The right argument N is the non-negative integer scalar or one-element vector specifying the number of terms of the sequence requested. The explicit result Z is a vector containing those terms. If N is 0, the empty numeric vector is returned. If N is a negative integer or not an integer, an error report is generated. Results are not rounded off, as the initial values and parameters may be any reals.

TRIB3 is a literal translation in APL of the algorithm for the calculation of the roots of the auxiliary or characteristic equation, and of the algorithm for the calculation of the Tribonacci numbers from those roots, as given in the two previous sections. Coefficients $x y z$ are set to zero when their calculated absolute value is smaller than an arbitrarily chosen tolerance. On an experimental basis this tolerance has been set to $1E^{-15}$. Calculation of the terms of the sequence from those coefficients may be simplified by using the outer product, but at considerable cost of performance in cpu time.

Standard ISO APL requires that for the dyadic function power A^*B , when the left argument A is a negative number and the right argument B is not an integer, a domain error report is generated, this to allow complex arithmetic to be a consistent extension. As some implementations conform to this requirement, we have coded line 19 of the function *TRIB3* :

```
[19] ΔP:UV+(×UV)×(|UV←-Q+(¯1 1)×Δ*0.5)÷3
```

instead of

```
[19] ΔP:UV+(-Q+(¯1 1)×Δ*0.5)÷3
```

Finally, if the value of T_n is only requested for one specific value of n , when applying recursion, the entire sequence $T_1 \dots T_n$ must be calculated. When applying explicit calculation from the auxiliary or characteristic equation however, T_n must only be calculated for that specific value of n . A modified version *TRIB4* of the defined function *TRIB3*, that only calculates T_n for a given scalar or one-element vector N is given below:

```

V Z+L TRIB4 N;P;Q;Δ;PHI;T;XYZ;UV;TR;TI;RHO;TETA
[1] +(N<3)/0,Z+L[N[3]
[2] P←((3×L[5])+L[6]*2)÷9
[3] Q←((27×L[4])+L[6]×(9×L[5])+2×L[6]*2)÷54
[4] +(0 1=×Δ+(Q*2)+P*3)/ΔZ,ΔP
[5] PHI+-20Q÷P×R+(-P)*0.5
[6] T←(L[6]÷3)+2×R*20(PHI+0,-1 1×02)÷3
[7] XYZ+(3+L)⊗qT°.*13
[8] XYZ+(1E-15<|XYZ)×XYZ
[9] →0,Z←+/XYZ×T*N
[10] ΔZ:→(Q=0)/QZ
[11] T←(L[6]÷3)+(-2 1 1)×Q*÷3
[12] XYZ+(3+L)⊗((13)°.*0 0 1)×qT°.*13
[13] XYZ+(1E-15<|XYZ)×XYZ
[14] →0,Z←+/XYZ×(1 1,N)×T*N
[15] QZ:T+L[6]÷3
[16] XYZ+((3+L)÷T*13)⊗(13)°.*0 1 2
[17] XYZ+(1E-15<|XYZ)×XYZ
[18] →0,Z←(+/XYZ×1,N,N*2)×T*N
[19] ΔP:UV←(×UV)×(|UV←-Q+(-1 1 1)×Δ*0.5)*÷3
[20] T←(L[6]÷3)+/UV
[21] TR←(L[6]÷3)-0.5×+/UV
[22] TI←(0.75×0.5)×|-/UV
[23] RHO←(×TR)×((TR*2)+TI*2)*0.5
[24] TETA←-10TI÷RHO
[25] XYZ+(3+L)⊗q(T*13),[1](2 1°.*0TETA×13)×2 3pRHO*13
[26] XYZ+(1E-15<|XYZ)×XYZ
[27] Z←+/XYZ×(1,2 1°TETA*N)×(T,2pRHO)*N

```

```

L+1 1 2 1 1 1
L TRIB4 10

```

149

```

(cL) TRIB4":10
1 1 2 4 7 13 24 44 81 149

```

The right argument is rounded off. If this rounded off value is less than or equal to 0, an error report is generated. Calculation of T_n from the coefficients $x y z$ has been simplified by using reduction, the impact on performance in cpu time being negligible.

Illustrations

Illustrations are given, using *TRIB3*, for all possible combinations of the signum of the discriminant Δ and of the signum of p and q :

a	b	c	Δ	p	q
2	1	-2	-	-	-
-1	-1	3	-	-	0
-2	1	2	-	-	+
2	-5	4	0	-	-
-3	5	-1	0	-	+
1	-3	3	0	0	0
1	1	1	+	-	-
-1	-1	2	+	-	+
2	-3	3	+	0	-
-1	-3	3	+	0	+
2	-2	1	+	+	-
4	-6	3	+	+	0
-2	-2	-2	+	+	+

```

L+1 2 3 2 1 -2
L TRIB3 10
1 2 3 -2 11 -18 43 -82 171 -338

```

```

L+1 2 3 -1 -1 3
L TRIB3 10
1 2 3 6 13 30 71 170 409 986

```

```

L+1 2 3 -2 1 2
L TRIB3 10
1 2 3 6 11 22 43 86 171 342

```

```

L+1 2 3 2 -5 4
L TRIB3 10
1 2 3 4 5 6 7 8 9 10

```

```

L+1 2 3 -3 5 -1
L TRIB3 10
1 2 3 4 5 6 7 8 9 10

```

$L+1$ 2 3 1 $^{-3}$ 3
 L TRIB3 10
 1 2 3 4 5 6 7 8 9 10

$L+1$ 2 3 1 1 1
 L TRIB3 10
 1 2 3 6 11 20 37 68 125 230

$L+1$ 2 3 $^{-1}$ $^{-1}$ 2
 L TRIB3 10
 1 2 3 3 1 $^{-4}$ $^{-12}$ $^{-21}$ $^{-26}$ $^{-19}$

$L+1$ 2 3 2 $^{-3}$ 3
 L TRIB3 10
 1 2 3 5 10 21 43 86 171 341

$L+1$ 2 3 $^{-1}$ $^{-3}$ 3
 L TRIB3 10
 1 2 3 2 $^{-5}$ $^{-24}$ $^{-59}$ $^{-100}$ $^{-99}$ 62

$L+1$ 2 3 2 $^{-2}$ 1
 L TRIB3 10
 1 2 3 1 $^{-1}$ 3 7 $^{-1}$ $^{-9}$ 7

$L+1$ 2 3 4 $^{-6}$ 3
 L TRIB3 10
 1 2 3 1 $^{-7}$ $^{-15}$ 1 65 129 1

$L+1$ 2 3 $^{-2}$ $^{-2}$ $^{-2}$
 L TRIB3 10
 1 2 3 $^{-12}$ 14 $^{-10}$ 16 $^{-40}$ 68 $^{-88}$

Benchmarks

Benchmarks have been done for $N = 250(250)1000$, in order to compare the effectiveness in cpu time for the functions defined.

Results are given below (in ms).

N	250	500	750	1000
TRIB1	97	219	365	564
TRIB2	68	155	262	390
TRIB3 ($\Delta < 0$)	7.1	11.2	15.3	19.5
TRIB3 ($\Delta = 0$ and $pq \neq 0$)	6.3	9.2	12.0	14.8
TRIB3 ($\Delta = 0$ and $p=q=0$)	4.8	6.6	8.5	10.3
TRIB3 ($\Delta > 0$)	9.0	14.3	19.6	24.9
TRIB4	3.8	3.8	3.8	3.8

On average, cpu times are 30% lower for *TRIB2* than for *TRIB1*. Depending on N and on the signum of the discriminant Δ the efficiency factor in cpu time is from 8 to 38 for *TRIB3* versus *TRIB2*. Figures are comparable to those reported for the Generalized Fibonacci or Horadam Sequences (1).

CPU times for *TRIB4* are based on a weighted average, as the impact of the signum of the discriminant Δ is negligible. The efficiency factor in cpu time is from 18 to 105 for *TRIB4* versus *TRIB2*, from 2.4 to 2.7 versus *TRIB3*. Efficiency factors are smaller than those reported for the Generalized Fibonacci or Horadam Numbers (1), due to the overhead while computing the discriminant, but are still considerable.

Functions defined conform to the standard ISO APL (11). Benchmarks have been done on a MicroLine Pentium S-100, with mathematical co-processor, using Dyalog APL/W Version 8.0.8 under Windows 95 (12). Benchmarks have been done using the defined function \square MONITOR.

References

1. J. De Kerf; *From Fibonacci to Horadam*; Vector Vol. 15 No.3, January 1999, pp. 122-130. See also: C.A. Reiter; *Exact Horadam Numbers with a Chebyshevish Accent*; Vector Vol.16 No.1, July 1999, pp. 122-131.
2. A.F. Horadam; *Basic Properties of a Certain Generalized Sequence of Numbers*; The Fibonacci Quarterly, Vol. 3, No. 3, 1965, pp. 161-176.
3. M. Feinberg; *Fibonacci-Tribonacci*; The Fibonacci Quarterly, Vol. 1, No. 1, 1963, pp. 71-74.
4. A. Scott, T. Delaney, and V.E. Hoggatt; *The Tribonacci Sequence*; The Fibonacci Quarterly, Vol. 15, No. 3, 1977, pp. 193-200.
5. C.K. Wong and T.W. Maddock; *A Generalized Pascal's Triangle*; The Fibonacci Quarterly, Vol. 13, No. 2, 1975, pp. 134-136.
6. J. De Kerf; *Some Other Triangles and APL*; Les Nouvelles d'APL, No. 30, Juillet 1999, pp. 71-99.
7. D. Jarden; *Third order recurring sequences*; In "Recurring Sequences" — A Collection of Papers; Reveon Lematematika, Jerusalem, Israel, 1966, pp. 86-89.
8. S. Rabinowitz; *Algorithmic Manipulation of Third-Order Linear Recurrences*; The Fibonacci Quarterly, Vol. 34, No. 5, 1996, pp. 447-464.
9. J.V. Uspensky; *Theory of Equations — Chapter V: Cubic and Biquadratic Equations*; McGraw-Hill Book Company, New York, New York, 1948; pp. 82-98.
10. U. Grenander; *Mathematical Experiments on the Computer*; Pure and Applied Mathematics Series; Academic Press, New York, New York, 1982; pp. 238-239.
11. International Standard ISO 8485 : 1989 (E), First Edition; 1989-11-01; *Programming Languages — APL*; International Organization for Standardization ISO, Geneva, Switzerland, 1989.
12. *Dyalog APL/W Reference Manual — Version 8*; Dyadic Systems Limited, Basingstoke, Hampshire, United Kingdom, 1996.

Joseph De Kerf
Rooienberg 72
B-2570 Duffel
BELGIUM

Onward to Viking APL in November ...



As APLBerlin 2000 progressed, it became increasingly clear that the north European contingent wanted a chance to get together again in a smaller group for one to two days of more focussed technical seminars.

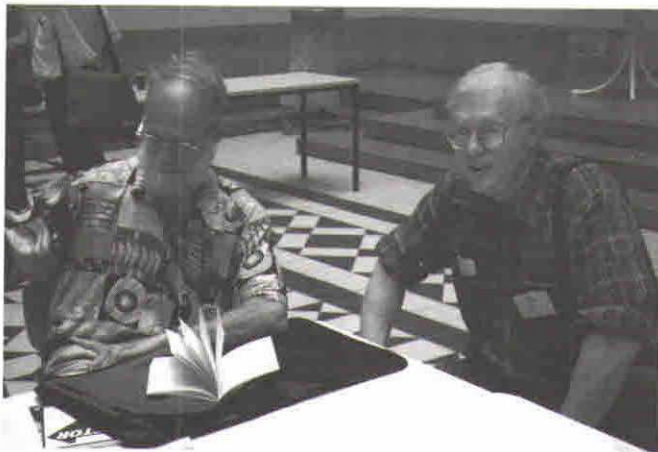
This fitted well with Insight Systems desire to re-establish themselves in the APL world, so the idea of a Viking APL meeting was born, with Gitte (on the right) and Helene (on the left of picture) as the organising committee.

It will be interesting to see how this goes - at the time of writing there are over 35 delegates booked to come, and this is always the kind of gathering where real work gets done and APL

as a whole moves a little way forward. Vector (as usual) will be there.

And to Yale in 2001 ...

Although nothing was announced at the conference, it now looks like Yale, Ed Shaw, and possibly June rather than August. Look out for firm dates on the SigAPL site as soon as they fix the time, place and committee.



Eke Van Batenburg with Ed Shaw

Index to Advertisers

Dyadic Systems Ltd	6
Riskflow	4
Strand Software	2
Vector Back Numbers	24

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 normally 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: _____

Category (please tick box) to run from: 1st May August Nov Feb

UK private membership £12
 Overseas private membership £14
 Airmail supplement (not needed for Europe) £4
 UK Corporate membership £100
 Corporate membership overseas £135
 Sustaining membership £430
 Non-voting UK member (student/OAP/unemployed only) £6

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

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.

Causeway Graphical Systems Ltd
The Mallings, Castlegate,
MALTON, North Yorks YO17 7DP
Tel: 01653-696760
Fax: 01653-697719
Email: sales@causeway.co.uk
Web: www.causeway.co.uk

Compass Ltd
Compass House
80 Priestley Road
GUILDFORD, Surrey GU2 5YU
Tel: 01483-514500

Dyadic Systems Ltd
Riverside View, Basing Road,
Old Basing, BASINGSTOKE,
Hants, RG24 0AL
Tel: 01256-811125
Fax: 01256-811130
Email: sales@dyadic.com
Web: www.dyadic.com

HMW Trading Systems Ltd
Hamilton House,
1 Temple Avenue,
LONDON EC4Y 0HA
Tel: 0870-1010-469
Email: HMW@4xtra.com

Insight Systems ApS
Nordre Strandvej 119G
DK-3150 Hellebæk
Denmark
Tel: +45 70 26 13 26
Fax: +45 70 26 13 25
Email: info@insight.dk
Web: www.insight.dk

APL2000
Rapid Application Development
6610 Rockledge Drive
Suite 502
Bethesda MD 20817
USA
Email: sales@apl2000.com
Web: www.apl2000.com

Soliton Associates Ltd
Groot Blankenberg 53
1082 AC Amsterdam
Netherlands
Tel: +31 20 646 4475
Fax: +31 20 644 1206
Email: sales@soliton.com

Dutch APL Association
Postbus 1341
3430BH Nieuwegein
Netherlands
Tel: +31 347 342 337

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.

2000/2001 Committee

Chairman	Adrian Smith 01439-788385 apl385@compuserve.com	Brook House Gilling East YORK YO62 4JJ
Secretary	Anthony Camacho 0117-973 0036 acam@tesco.net	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	Ray Cannon 01252-874697 100430.740@compuserve.com	21 Woodbridge Road, Blackwater, Camberley, Surrey GU17 0BS
Activities	Jon Sandles 01904-612882 jon_sandles@csi.com	138 Burton Stone Lane, YORK YO30 6DF
Education	Dr Ian Clark 07931 370304 earthspot@aol.com	1, Heifer Mill Cottages, Mosterton, Beaminster, Dorset DT8 3HG.
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, Ray Cannon (01252-874697), Marc Griffiths, Bob Hoekstra (01483-771028), Jon Sandles (01904-612882)	

Typeset by APL-385 with MS Word for Windows
Printed in England by Short-Run Press Ltd, Exeter