VECTOR

APL Tools and Techniques ...

- Clark on Defined Operators 61
- BAA Vendor Forum 67
- An APL Unicode Font 75
- Windows BMP Files and J 95
- Reiter on Horadam Numbers 122
- A J/Lapack Interface 133

The Journal of the British APL Association

A Specialist Group of the British Computer Society

ISSN 0955-1433
www.vector.org.uk
Vol.16 No.1 July 1999

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 1999-2000

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, Brook House, Gilling East, YORK YO62 4JJ. Tel: 01439-788385 Email: apl385@compuserve.com

Contents

		Page
Guest Editorial: The Walls are Closing in ...	Adrian Smith	3
APL NEWS		
Quick Reference Diary		5
Correspondence		7
British APL Association News		12
APL99 Abstracts		20
APL Product Guide	Gill Smith	25
Microsoft Announce New APL		39
The Education Vector		
Zark Newsletter Extracts	edited by Jon Sandles	42
J-ottings 21: Time for Amendment of Data	Norman Thomson	56
What Use Are User Defined Operators?	Ian Clark	61
RECENT MEETINGS		
BAA Vendor Forum	Stefano Lanzavecchia	67
GENERAL ARTICLES		
An APL Unicode Font	Phil Chastney	75
GUI and Closed Systems	Anthony Camacho	86
Namespaces: New Gem from Old Roots	Alexander Balako	92
Windows BMP Files and J	Brook Anthony Schofield	95
TECHNICAL SECTION		
Hacker's Corner: A Special Locator in Dyalog APL/W	Joachim Hoffmann	108
Technical Correspondence		110
At Play with J: New Big Deal	Gene McDonnell	113
Purging Name Pollution in APL+Win	Ajay Askoolum	120
Exact Horadam Numbers with a Chebyshevish Accent	Clifford A. Reiter	122
J/LAPACK Interface...	Procter & Brown	133
Index to Advertisers		143

Powerful Enterprise Software Solutions for Today and Tomorrow.

Soliton Associates provides high-performance enterprise-wide software systems that address today's business needs. Combining expertise in the information technology business with quality software products and a full range of technical support, Soliton provides solutions that inform, connect, and grow. And Soliton's continual investment in research and development ensures that its products keep pace with the ever-changing face of technology.

Soliton products are aimed at data-intensive and time-critical problems across all industry segments.

TIMESQUARE A Timeseries-Relational Database Management System for acquiring, managing, and disseminating historical financial market data for the enterprise.

VIEWPOINT A powerful, flexible, easy-to-use data manager and report generator for all kinds of data

SHARP APL A highly productive application development environment for MVS and UNIX platforms

For information call the Soliton office nearest you. Send e-mail to sales@soliton.com or visit our web site at www.soliton.com.

SOLITON ASSOCIATES
Enterprise Software Solutions

TORONTO	NEW YORK	AMSTERDAM
416.364.9355	212.344.9388	+31.20.646.4475

Guest Editorial: The Walls are Closing in ...

by Adrian Smith

Stef is taking a well-earned break in Italy, so Vector Production is taking its chance to trail some of its wilder ideas and fears across the bows of the APL community. Firstly, I must thank both our new Editor and the rest of the working group for taking the deadlines seriously, and getting Vector back on schedule. This time, we had plenty of good material to choose from, and it was all assembled and ready for typesetting by the beginning of June. In fact there is a lot of good stuff already lined up for 16.2; this makes it so much less hassle to get the final copy prepared, and allows us to get much of the material on the Vector web site in parallel with the physical publication.

One clear consequence of keeping the web site well maintained and indexed has been a rise in demand for back numbers. Today we parcelled up the 3rd complete set of back numbers in as many months, and there has been a steady trickle of requests for odd volumes. Yes, the stock is still there to do this, but only just! We are down to single-figure stacks of several early editions, so if you want to complete your set, better get in touch soon!

On to the fears and hopes! Did you ever drive up one of those narrow lanes in the Yorkshire Dales where the walls seem to be closing in on you? The effect is most marked late at night, and the experience is enhanced when you come upon a herd of Aberdeen Angus (the black, woolly ones with green eyes) in the middle of the road. What has this to do with APL? In particular, what has it to do with the APL character set and the Internet? Consider for a moment what we are asking the browser to do ...

- <code>2+2</code> tells it that what lies between these tags is computer code, so use a fixed pitch font and don't muck about with silly tricks like smart quotes. In IE5 it is *essential* to use these tags, unless you want results like `∇f∆ ⎕WC ∇Form∆`, or are happy to see all your subtraction signs rendered as ±.
- <font face="APL2741"> says 'use this font please'. In IE4 all was fine, but in IE5 you discover that unless the font is marked as 'symbol encoded' any characters in the range 128-160 are rendered with Courier as the browser assumes that the font you requested cannot produce them! This makes APL code pretty unreadable, as a quick glance at Charmap will confirm!

So far, not too serious, until you set the appropriate switch to get it all working in IE5, sit back contentedly, and drive slap bang into the back of Netscape-4 which cannot get its tiny brain around the notion of a symbol-encoded code font, and

refuses to use it at all! *It really is all our fault* ... computer code uses the low ASCII characters, everyone knows *that!* It's not that the browser-writers are out to get us intentionally, it's just that we are so far outside their mindset that the walls they are building are closing in fast and there may already be no escape.

So much for reproducing the character set, now what about the opposite problem — typing it in to an editor! Just how many of us old mainframers have to die before someone finally has the courage to dump the old 2741 layout and make a rational mapping of symbols to keytops? OK, we learned to type APL when there was much less to learn — you got to know the set of base symbols quite quickly and the funny ones were composed with judicious use of the backspace key. OK, {jot}backspace{shoe} was a bit bizarre for {comment}, but in general you got all the common stuff in single keystrokes and many were quite easy mnemonics.

Then we had the old 3270 screens, with a pretty accidental scatter of overstruck symbols generated by various Alt+key combinations. At least we still had them engraved on the key tops! On to APL*PLUS, and the random scatter was legitimised as the 'union' keyboard — Dyalog even preserved the mapping of quote to Ctrl+K, but at least they gave us *stickers*. Now we have the worst of all possible worlds — a stupid mapping and absolutely zero assistance from the keytops or the APL environment.

And you wonder why no-one is learning APL any more.

Quick Reference Diary 1999–2000

Date	Venue	Event
August 10-14 1999	Scranton, Penn.	APL99
Late July 2000	Berlin	APL'00

The APL99 Conference is sponsored by SIGAPL and NYSIGAPL and is now in cooperation with ACM/SIGPLAN. The scope of the conference will be extended to include all array-processing languages including Mathematica, MATLAB, Array Based Fortran, SISAL and SAC. SIGPLAN members will be able to attend the conference at member rates. We are grateful to acknowledge the Arthur J. Kania School of Management at the University of Scranton for providing the conference facilities.

An electronic mailing list for information about APL99 has been set up. Subscription information is available on the web site

http://www.acm.org/sigapl/apl99.htm

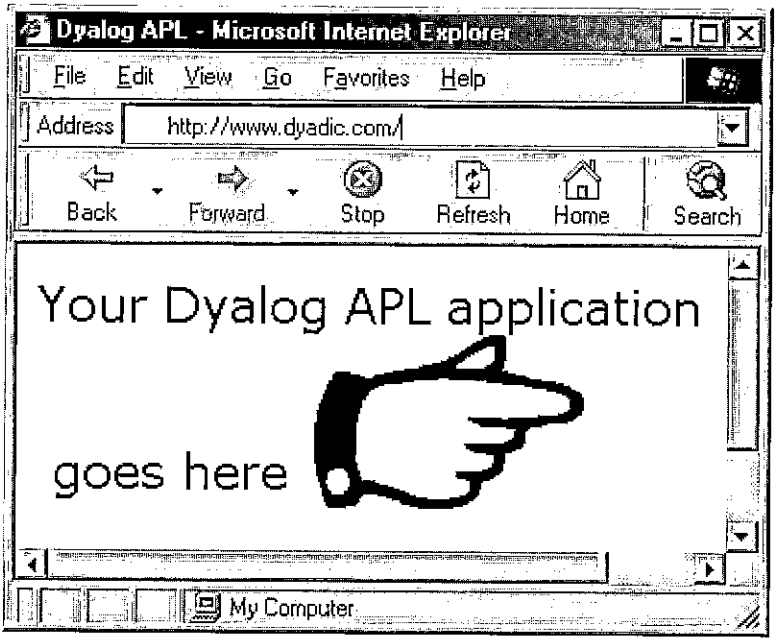
We are tentatively planning to schedule pre-conference tutorials on Tuesday, August 10, 1999.

Dates for Future Issues of VECTOR

	Vol.16 No.2	Vol.16 No.3	Vol.16 No.4
Copy date	10th September	4th December	10th March 00
Ad booking	17th September	11th December	17th March
Ad Copy	24th September	18th December	24th March
Distribution	October 99	January 2000	April 2000

dyalog APL

The Definitive APL for Windows ™



Using Dyalog APL, not only can you write a powerful, fully functional, **multi-threaded** Web Server, but you can also write ActiveX controls that automatically download and run in the Web browser.

Dyalog APL - the Internet - the Future

Dyadic 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

CORRESPONDENCE

From: Bob Hoekstra March 1999

Finally it is happening!

Many of us have been predicting this for years, although the more pessimistic among us may have doubted that it would ever really happen. What am I burbling about? *The Rise of Linux* of course. The recent flush of news articles caused my pulse to skip a beat, but the crowning glory is IBM's announcement of *support*. For those who missed it, please point your web browser at http://www.sunworld.com/swol-03-1999/swol-03-linuxworld.html, but for the lazy I'll quote the juiciest bits:

> "San Francisco (March 3, 1999) — This week at the LinuxWorld Expo in San Jose, IBM detailed its plans for Linux, including extensive, seamless, single point-of-contact global support to be offered in conjunction with the four major Linux vendors, Red Hat, SuSE, Caldera, and Pacific HiTech ...
>
> IBM also announced its plans to port several IBM products to Linux ... and a port for Linux to certain RS/6000 models, including the high-end POWER3-based 43P Model 260 ..."

Your readers may well want to know why I am writing about this to an APL publication. Fortunately I have a very good reason, and (I believe) a convincing argument. Please read on.

Many times I have heard APLers whinging about the apparent decline in APL usage and the fact that many APL projects seem to be floundering. Personally I try to ignore most of this, but there is no denying that APL is not doing as well as it deserves to be. Many times I have asked myself if there is a good reason for this, and the only real reason I can come up with is "lack of support" (some readers may cry "Obvious!" at this point, but please bear with me).

The first time I became aware of this phenomenon was when I noted (I think it was in 1986 or 1987) that APL was not on a list that IBM sent to clients describing "strategic software" products (I think that was the phrase used). I asked about this and was told that APL had not been (could not be?) ported to the AS/400 platform and hence IBM was not going to market our beloved language very actively. To any who think that IBM's best marketing efforts aren't worth much (*e.g.* the spectacular lack of success that OS/2 has enjoyed) I would like to point

out that the PC itself only really took off because IBM produced it first. Please don't tell me about the importance of the spreadsheet in the development of the PC. Spreadsheets were also introduced on other small computers, some arguably more advanced (better) than the PC, but nobody remembers them. A blessing from IBM was (and probably still is) critical for a new product.

IBM is no longer the force it used to be. Nevertheless, people still tend to sit up and take notice when Big Blue says something. They may disagree, or even ridicule, but they notice. This endorsement of Linux is (in my opinion) the most important step thus far to what I hope will be near world domination for this excellent operating system. Please don't get me wrong, I don't want to convert all the readers to Linux with this letter (though that would be nice as a longer term goal). Instead I want to alert your readers to the fact that there is no good (not even adequate) APL interpreter for Linux.

THIS IS A BIG, **BIG** MISTAKE!

Nobody knows how many Linux users there really are, but I have heard estimates exceeding 12 million, with 7 million apparently being a reliable lower estimate. Compare this to around 11 million OS/2 licenses sold. Until recently they were the loony types (like me) who wanted to get away from MickeySoft and run a real OS at home, but this is changing. Big companies are installing free operating systems (Linux, FreeBSD, NetBSD and others, with Linux being by far the fastest growing OS) and running them on critical systems. If you don't believe me, search the web. They are developing many, many applications in C(++), Java, and many other languages, but not APL.

These people need an APL. Even the loony types (like me) want an APL. Not something like APL/II, brain dead and without a decent APL character set, slow, non-standard and ancient, nor J (which is available) as I think this initial learning curve is far steeper than for a conventional APL. They want a modern, fully featured APL with nested data structures and a good X11R6-friendly interface. They want something that will produce applications quickly using the power they have to hand. They have an efficient operating system on powerful machines — *i.e.* APL heaven!

Linux is going from strength to strength. In 1998, 17% of new servers got Linux as their operating system. Pledges of support have come from all sorts of sources, including IBM/Lotus, Sun Microsystems, Hewlett Packard, Silicon Graphics, SCO, Oracle, Sybase, Corel and more. IBM have indicated this as an alternative to AIX on RS/6000 machines (note that they are doing the port themselves!) and Sun have been giving critical support to the developers of SPARCLinux for the

UltraSPARC processor (as an alternative to Solaris 7). This is serious support — these companies are not playing around!

Many implementations seem to be heading for the desktop, not just the server. APL might be able to buck this growth, but I believe this could be fatal. There is a huge market of computer fanatics out there who would benefit from APL (and vice versa) but they cannot. I believe that the interpreters ported to Linux will benefit greatly in high volume sales over the coming years. Those that don't will have high cost, low volume sales which will become increasingly difficult to justify in a corporate context and impossible to justify in a personal/home context.

More directly, I am convinced that Dyadic Systems, APL2000, IBM and Soliton (and any others) could port their UNIX APL interpreters to Linux with little effort. Economic arguments against this don't hold water. In the long term, they must do it. I would like to suggest that these companies try this with an old version. For instance, Dyalog APL/X version 6.x would be interesting to port (perhaps even released as a no-support freebie) to see the response for a relatively small capital cost.

I'd like to close with a quote from another SunWorld article

(http://www.sunworld.com/swol-03-1999/swol-03-sco.html):

> "...Today, any company that doesn't include Linux somewhere in its roadmap is missing the boat..."

Please let us see APL grow and prosper.

Bob Hoekstra
Bob.Hoekstra@khamsin.demon.co.uk

P.S.: In case it's not obvious, I feel quite strongly about this.

From: Anne Wilson 6 June 1999

Do I have to use Recursion?

My problem started when I tried to do the *Limbering Up* exercise in Vector 15.4. All went well until I met the line "Oh, one more thing, TREE must be recursive. That is, it must call itself." By the time I read this I had mentally solved the problem, and nearly finished coding it, allowing for recursive data but without the use of recursive programming. Why should TREE be recursive, and whatever will these recursive calls do? My other problem was the variable shown; because my code is not recursive I have no use for such a variable. Is there a hidden assumption here that because the data is recursive it requires a recursive function?

As a professional programmer, and proud to be one too, I have made a study of program design. My aim has always been to produce the simplest program that will solve the problem. Sometimes one starts to write code and thinks *"There must be a simpler way to solve this"*, and starts again finding the simpler method, usually by deliberately designing before starting on the code writing.

There are three main components in a program — input data, output data and processing. If the program design mimics the structure of one of these components then the programming is not adding extra complexity; if it mimics the simplest of these structures then it is producing the simplest program. (Obviously the programmer is always aware of the other components when making design decisions.) From this it follows that a program only needs to be recursive if all three components are recursive.

But before reaching this stage the program designer should see if a restatement of the problem will result in a simpler program. An obvious example is the summation of a geometric progression where the problem is usually stated in a recursive way. The recursive method calculates the next term, and then calls itself, returning with the accumulating sum; the accuracy is determined by a criteria embedded in the code. However, if you know the formula for the sum all that is required is one calculation; as an added bonus the answer is absolutely accurate (except for limitations of the machine). This requires the inclusion of knowledge, which may be unavailable to the person requesting the program.

In my measurement of complexity recursion is high, looping medium and a single calculation is lowest and hence most desirable. In APL terms looping should include any function that is hiding a generated loop, such as *each* and some *outer products*. Sometimes it can be demonstrated that looping or recursion

will be required, because some of the input data can only be determined after earlier data has been processed.

I solved the *Limbering Up* with a loop to find the order that boxes will be shown, and their depth in the diagram. After that I used a single calculation to place the names, boxes and lines onto the output diagram. But I have also effectively restated the problem as I have knowledge that all tree relationships can be described by the *depth* of each box in the diagram and its position in a *traipze* round the tree diagram; and that once the data is in this format all the processing can be done in single statements without recourse to loops or recursion. Note that a true mathematical tree cannot be recursive — but that tree structures can be used to describe data that is recursive.

Now you have probably already thought what a simpleton I must be to think programs are so simple! Many of the nasty complications arise at the input and output stages, and they just have to be coded in but do not affect the structure of the program; others are due to conditional processing; but all is not lost even then if an OOPS approach is used to isolate these complications. Where a program performs several functions each one should be treated separately, so that a large system to input data, perform BOMP calculations, produce reports, etc. etc. will have each component designed separately, with the overall system design being another part of the design.

British APL Association News

Annual General Meeting

21 May 1999 at the RSS, beginning at 2 pm

Minutes of the AGM

Apologies were received from Dr Alan Mayer and Dr Ian Clark.

The minutes of last year's AGM (published in Vector Vol.15 No.1) were taken as read.

The Chairman reported on the year (full report below). He explained that as no post had more than one candidate there had been no need to hold a ballot, but the committee would be very glad to hear from anyone who wanted to help and such people could be co-opted to join the committee.

The Treasurer circulated the accounts (see below) and asked for questions: there were none.

There were no questions to the committee.

John Sullivan was re-elected as auditor.

The meeting ended at about 2.15 pm.

Chairman's report on the BAA year 1998-1998

In short; we have held two meetings and Vector has (approximately) appeared on time.

Your committee as announced at the last AGM was:

Chairman	Anthony Camacho
Secretary	Ajay Askoolum
Treasurer	Nicholas Small
Editor	Vacancy
Projects and publicity	Dr Alan Mayer
Activities	Jon Sandles
Webmaster	Ray Cannon

We had a vacancy for Editor. Stefano Lanzavecchia had been guest editor for the April 1998 Vector. You will be delighted to hear that he agreed to take on the task for another two years. As he lives in Denmark, goes home to Italy for some of his holidays and frequently visits the UK as he works for Adaytum, we bought a portable computer for him to use and keep in constant touch with the other members of the Vector working group and with the production office. This he has done and I hope you will agree with me that under his care Vector has maintained or improved the standard set by previous editors.

The two meetings were the vendor forum after last year's AGM and the visit Eric Iverson made on his way back from APL 98. Both meetings have been documented in Vector so I will say no more now except to thank Jon Sandles for arranging them.

Next year's committee will be:

Chairman	Anthony Camacho
Secretary	Ajay Askoolum
Treasurer	Nicholas Small
Editor	Stefano Lanzavecchia
Projects and publicity	Dr Alan Mayer
Activities	Jon Sandles (co-opted)
Webmaster	Ray Cannon

Membership continues its slow decline; we now have 441 members. The circulation of Vector is 628 copies. Over the years the committee has tried many things to encourage people to join the Association and discourage people from leaving. We would like to hear from any members who think they can do something to reverse this trend. Our email addresses and telephone numbers are in Vector. It is in every member's interest to recruit more members. A larger audience makes it possible to attract speakers for meetings and a larger readership motivates the Vector working group to keep the magazine going.

The Association is financially sound. Profits from conferences are still unspent and we have about £45,000 in the bank. I feel entirely happy that our expenditure exceeds our income by three or four thousand pounds a year and so we do not propose to change the subscription rates. At current expenditure our money will last longer than our members. I would be glad to increase our deficit if, by spending the money on some well-designed recruitment initiative, we could increase our membership.

I conclude by thanking your committee and the Vector working group for their efforts which have kept the Association going over the last year, in particular those who have done significant work and are not mentioned above - Nicholas

Small, Ray Cannon and Ajay Askoolum. Lastly I thank the authors whose contributions to Vector are the main reason why new members join and why old members renew.

Anthony Camacho, 20 May 1999

Accounts 1998/99

from Nicholas Small (Hon. Treas.)

Notes to the Accounts

1. Pence figures have been omitted, so columns may not add exactly.
2. The value of stocks of Vector and the Software Library have not been assessed, nor has the value of the Association's computing hardware and software.
3. For 1998/99, figures are shown both as income and expenditure, i.e. revenues strictly relating to the activities of that year, and as receipts and payments, *i.e.* what goes in and out of our bank account. The comparitive figures for earlier years relate to income and expenditure.
4. Note that from 1997/98 VAT collection payments are made net of any rebate, so VAT rebates no longer appear under Income/Receipts, Other.
5. The sum written off comprises a bad debt of £500 (Uniware) and a cancelled invoice, half of £135 (Realkredit).

In comparing with last year allowance must be made for the purchase of the laptop (with insurance, about £2000 against Vector) and for the fact that we actually managed to hold two meetings. Interest rates have fallen again, as has the sum on which interest is payable; these effects were mitigated by the lateness of Vector 15.1 and 15.2.

There has been another significant drop in membership and Vector circulation.

British APL Association — Summary of Annual Accounts 1998/99

Summary of income and expenditure/receipts and payments:

	1998/99 (R&P) £	1998/99 (I&E) £	1997/98 (I&E) £
Income/Receipts			
Subscriptions	10420	10368	11021
BCS services	167	167	0
Bank interest	2943	2943	3322
Vector advertising (incl.VAT)	2963	2463	3471
Other	264	264	108
Total receipts	16757	16206	17922
Expenditure/Payments			
Meetings	704	704	0
Administration	1471	1423	1152
BCS services	167	167	0
Vector production and despatch	17337	17086	14584
Education Vector supplement	-	-	108
Projects	328	328	501
Other	221	246	308
Total payments	20228	19954	16653
Assets summary:			
Bank and other balances	49055	52525	
Debtors	733	2071	
Creditors	(4776)	(5270)	
Net assets	45012	49327	
Written off	(568)		

APL Keyboards for VECTOR Subscribers!

by Ian Clark

Very few APLers enjoy the luxury of using a keyboard having proper APL characters on the keytops, especially matching their own choice of keyboard layout. This is one reason why it is so inconvenient to teach APL to a class of students using centralised facilities. It also causes problems for consultants having to debug APL code on-site, on someone else's machine, even. Being on-site in a foreign country has its own problems (I frequently have to use a keyboard in Denmark), where you even have to hunt for the '£' sign, not to mention '@' and '[' and ']'.

Now everyone has their own favourite solution to this problem, but it's safe to say that there's no general, complete solution on the horizon. Various software solutions have been proposed. APLs have the ability to re-map keyboard layouts, but to re-map somebody else's keyboard isn't the work of a moment and will earn you no great deal of thanks.

APLomb (Clark, 1992), designed for educational use by APL novices, has a floating palette with the commoner symbols, converts '==' into {gets} on-the-fly, and offers a sort of wizard to identify and type the symbol you need, even if you can't quite remember what it is. And of course there are always the various systems of ASCII keywords, like {comma-underbar}... Wunderbar! In desperation you can type {quad}AV (if you can find {quad}) and copy/paste the output.

I've been looking into hardware solutions, *viz.* to be able in effect to carry your own labelled keyboard around with you. These might be attractive to firms setting up new APL programming shops (yes, there are some...!) or to schools and colleges considering serious classroom use of APL. Just at the moment employment prospects for APLers in UK and Europe aren't too bad, so maybe it's not totally out of the question to contemplate teaching APL again.

Contech, based in Stevenage, will engrave keytops of standard IBM-compatible PC keyboards with the symbols of your choice, in a range of colours, if you provide them with your requirements in the form of simple artwork (a drawing will do). They will either supply you with a new low-cost standard IBM-compatible 'Cherry' keyboard, suitably engraved, or will take your existing keyboard and engrave the keytops. These must be able to pull off, so this seems to preclude laptops. Contact Suzanne with your requirements on +44 (0)1438 315757 (fax: +44 (0)1438 313679, email: sales@contech.co.uk).

What does it cost? I phoned Suzanne, explained that an APL 'unified' keyboard would require 48 to 50 keys to be engraved and asked for some cost estimates. A single keyboard would cost £55, but with an order for ten of these the cost would come down to around £30 per keyboard. Add £17 for the low-cost 'Cherry' keyboard if you want the firm to supply the keyboards too. Allow 10-14 days for the order to be completed.

Another solution is to use a 'keyboard glove'. This is a close-fitting flexible transparent cover, usually to protect a keyboard in a dusty or coffee-laden environment, but the glove can be fitted with keytop labels, preferably on the underside.

Inpace Ltd manufactures keyboard gloves, using your own keyboard if necessary (1 day turnaround excluding delivery times) but has some 2,000 moulds in stock, covering all the common brands of keyboard. Gloves for any type of keyboard cost the same, depending on the quantity ordered: £19.95 for single gloves, £14.95 each for 5, £13.95 each for 10. The firm uses an outside supplier to print keytop labels. As with all printing, costs come down with volume, so that a print-run of 1,000 would cost just £1.90 per set of labels. Add to this £2.00 per keyboard glove for the labour to stick them on. Get in touch with Ben with your requirements, on +44 (0)1993 706303.

Now it isn't an attractive proposition to pay for a minimum order of, say, 50 sets of labels if you only want one keyboard glove. We discussed the possibility of the British APL Association paying for a print-run, which would be available free to members, *or their organisations*. You would then simply phone Ben, tell him you're a bona-fide VECTOR subscriber, and explain to him the layout you want or ask for one of the standard ones, and pay only the £2.00 fitting charge on top of the basic charge for a keyboard glove. Since each keyboard is hand-labelled, you can have any label on any key. The simplest and most reliable way to specify the glove is to photocopy your keyboard and write the symbols on the photocopied keytops by hand. This, plus the make, model and FCCID number, is enough to identify the keyboard too. I'm told that laptops aren't entirely out of the question either.

In the interest of promoting the use of APL, which is part of the British APL Association's brief, we the committee are happy to collect requests and enquiries. If there's sufficient interest, we will order a trial print-run of keyboard stickers, either sticky-topped for applying to the underside of a keyboard glove as described, or sticky-bottomed, for applying directly to keytops. The intention is to furnish these as a free service to VECTOR subscribers, *and their organisations*, for their own use (but not for re-sale, of course). However there must be sufficient

people interested in order to go ahead. In a future issue we shall announce an ongoing policy, *i.e.* who's eligible and for how many, and whether you can pay to have more. Meanwhile, let's hear your requirements.

Contact any member of the BAA Committee (see this issue's back cover), including myself:

Ian Clark, IAC Human Interfaces, R21 Auckland Business Centre, St Helen Auckland, Bishop Auckland, Durham DL14 9TX, England. Tel: +44 (0)1388 605544. Mobile: +44 (0)7931 370304, Email: 100021.3073@compuserve.com.

APL99 Abstracts and Tutorials

Invited Speakers

High-Confidence Design for Security, Prof Shiu-Kai Chin. The widespread use of networks makes information security a major concern where the underlying network (e.g., the Internet) is assumed to be insecure. Systems with security requirements typically must operate with a high degree of confidence — they must be highly assured.

The task of designing and building secure systems raises a fundamental question, *how do we know with confidence that our designs will behave securely?* Having confidence in a secure system requires having confidence in the following: the strength of the cryptographic algorithms, the correctness of the hardware and software implementations, and knowing the implementation supports a security model.

This article describes methods that establish confidence that implementations meet their specifications and security requirements. These methods are rigorous in nature. They rely on mathematical logic and are accessible to engineering students at the masters level. As is typical in systems engineering, a variety of methods are used depending on what level of design is being addressed.

Professor Shiu-Kai Chin
Laura J. and Douglas L. Meredith Professor for Teaching Excellence, University, Syracuse
Email: skchin@syr.edu

An Overview of E-Commerce and Intelligent Agents in Supply Chain Management and What it Means to the IBM Micro-Electronics Division, Kenneth J. Fordyce, IBM Corporation. Since the middle of the 1990s one the hottest areas of Decision and Information Systems has been Supply Chain Management. At the end of 1997 another hot area began to emerge called E-Commerce. In September of 1998 the author and Garry Sullivan, IBM Burlington, Vermont, were given the assignment to evaluate E-commerce alone and its relationship to Supply Chain Management, develop a proposed plan how E-Commerce technology can be used in Micro-Electronics Division's (MD) supply chain applications and business process, identify some initial projects with MD in this area, and develop partnerships with Universities and the NSF. This presentation will provide an overview of their work to date.

Kenneth J. Fordyce
Email: FORDYCE@US.IBM.COM

Object-Oriented Concurrent, Parallel, and Distributed Programming in Java, Prof Doug Lea. An introductory survey of Java constructs and common designs used in concurrent, parallel, and distributed programming.

Main topics include:

- Approaches to design
- Threads, locks, and monitors
- Exploiting multiprocessors
- Distributed objects using RMI

Doug Lea is a professor of Computer Science at the State University of New York at Oswego. He is author of the book, *Concurrent Programming in Java*, and co-author of the text, *Object-Oriented System Development*. He is the author of several widely used software packages, as well as articles and reports on object oriented software development including those on specification, design and implementation techniques, distributed object systems, and software reusability.

Doug Lea, Computer Science Department
State University of New York at Oswego
Email: dl@cs.oswego.edu

The Switch Back Gravity Railroad and APL2, Walt Niehoff. A discussion of the gravity railroad that was America's second railroad and first coal-hauling railroad. It was the first to open up the Anthracite Region of which Scranton was a big part. The railroad was called the Switch Back Gravity Railroad and was based at Mauch Chunk, Pennsylvania (now Jim Thorpe) about an hour south of Scranton. As an early user of APL since the late 1960's who had co-authored GRAPHPAK, the APL graphics package, during his days at IBM, Walt used APL2 and GRAPHPAK in his research on the gravity railroad. (http://www.software.ibm.com/ad/apl/apl2-sbgrr.html)

Walt Niehoff is the author of *The Reincarnation of Switch Back Gravity Railroad*, 1995. While at IBM, he co-authored GRAPHPAK, a graphics package written in APL.

Walter H. Niehoff, Consultant
Email: niehoff@spectra.net

Abstracts of Papers

Generalization of Pick's Theorem for Surface Polyhedra, Dr. Mihaly Agfalvi, Istvan Kadar and Erik Papp, Budapest, Hungary. Because Pick's theorem was first published in 1899, our planned presentation is timed for its 100th anniversary. Currently it has greater importance than realized heretofore because it forms a connection between the old Euclidean and the new digital (discrete) geometry. Today the question is not the old one: how can we produce traditional area without co-ordinates, using only inside points and boundary points. Just on the contrary: how is it possible to simply determine digital boundary and digital area (namely the number of boundary points and inside points) using known co-ordinates of vertices.

Dynamic Systems Simulation Using APL2, Robertas Alzbutas and Vytautas Janilionis, Kaunas, Lithuania. In this paper, we describe methods, models and software applied for dynamic system simulation using APL2.

When Bears are Blue and Bulls are Red, Linda Alvord and Tama Traberman, Milford, NJ. Analysts deal with real numerical data. Often the data is collected over time. Because numbers are abstract, analysts often use graphs to make numerical data more meaningful. Real data typically fluctuates in some random fashion. However, patterns often exist in the data. An example would be the seasonal fluctuations in commodities. Analysts use a variety of techniques to help them find patterns that have the potential to improve their skill in predicting future trends. This paper suggests some ways to make some of these patterns more obvious.

Functions and Data Can Dance as Equal Partners, J. Philip Benkard, Stamford, CT

The Zark Library of Utility Functions, Gary A. Berquist, Ellington CT. The primary appeal of APL has always been its PRODUCTIVITY. You can develop and maintain computer applications faster with APL than with any other high-level programming language. APL has a number of distinct traits that contribute to this productivity advantage over other languages. However, language productivity is only one side of the programmer productivity triangle. The other two are programmer experience and utility software. The programmer with the most productive language, the most experience, and the best utility software will develop better code, and faster than the programmer who is deficient in any of these three areas. Experienced APL programmers generally score well in the first two areas, but poorly in the third, probably because APL obviates the perceived need for high quality utility software. Programmers in other languages do not have this luxury and so tend to develop and rely upon more extensive and higher quality utility software. Consequently, the productivity advantage of APL programmers over other programmers is not as significant as it can be. ZarkLib, the Zark Library of Utility Functions, is a scheme that attempts to remedy this problem. It is both an extensive collection of APL utility software and a motivational scheme. It guarantees its growth and maintenance by employing a "pyramid" scheme that motivates APL programmers to give until it hurts, and then give again.

Choices in Server Side Programming, A Comparative Programming Exercise, Robert G. Brown and Willi Hahn, Lingo Allegro, Chicago, IL

Regions: An Abstraction for Expressing Array Computation, Bradford L. Chamberlain, E. Christopher Lewis, Calvin, and Lawrence Snyder, U of Washington, Seattle, and U of Texas, Austin. Most array languages, including Fortran 90, Matlab, and APL, provide support for referencing arrays by extending the traditional array subscripting construct found in scalar languages. We present an alternative to subscripting that exploits the concept of regions — an index set representation that can be named, manipulated with high-level operators, and syntactically separated from array references. This paper develops the concept of region-based programming and describes its benefits in the context of an idealized array language called *pure*. We show that regions simplify programming, reduce the likelihood of errors, and enable code reuse. Furthermore, we describe how regions accentuate the locality of array expressions and how this locality is important when targeting parallel computers. We also show how the concepts of region-based programming have been used in ZPL, a fully-implemented practical parallel programming language in use by scientists and engineers. In addition, we contrast region-based programming with the array reference constructs of other array languages.

Accelerating APL Programs with SAC, Clemens Grelck, Sven-Bodo Scholz, University of Kiel, Germany. The paper investigates how SAC, a purely functional language based on C syntax, relates to APL in terms of expressiveness and run-time behaviour. To do so, three different excerpts of real world APL programs are examined. It is shown that after defining the required APL primitives in SAC, the example programs can be re-written in SAC with an almost one-to-one correspondence. Run-time comparisons between interpreting APL programs and compiled SAC programs show that speedups due to compilation vary between 2 and 500 for three representative benchmark programs.

Sparse Arrays in J (Preliminary Version), Roger Hui, Toronto, Canada

INFO: Interactive APL Documentation, George Mebus, Lex2000, Inc. Princeton, NJ. A large body of APL code may be hard to understand and analyze, particularly if you are not its author. A code system that spans multiple workspaces (WSs) compounds that problem.

INFO is a Multi-WS system written in APL+Win that provides convenient interactive documentation of

APL+Win and APL+DOS multi-WS systems. The User has a variety of displays that give insight into the system structures and relationships within single- or multi-WS systems. An administrator can easily set up and maintain the INFO static analysis data bases for any number of WS groups.

This paper demonstrates INFO by showing how INFO documents itself. A distribution package contains the complete code and instructions for setting up this self-documentation.

A Retro/Prospective on APL GRAPHPAK. Walt. Niehoff, Vestal, NY. This paper suggests two general directions that one could take to modernize APL2 Graphpak and revitalize its evolution. One direction springboards off lessons learned during Graphpak's first decade and from some thinking that evolved during early (c. 1980) experiments with general arrays. The second direction exploits general arrays and APL2 functionality at a user-level. Some experiments are reported relating to both areas.

Teaching J as a Computer Notation for Secondary Mathematics, Howard A. Peelle, Amherst, MA. This article summarizes a teacher-education course which introduces J as a computer notation well-suited for teaching secondary mathematics. This one-semester course is designed as a series of workshops with accompanying discussions. The first workshop is described here in detail; the others are sketched. Teachers' experiences and reactions to using J are reported informally throughout.

An Object-Oriented Approach to Educational Software in Building Physics, G. Reichard, Graz, Austria. An object-oriented building-physics software A.T.O.N. (which is the acronym for the German "General Thermal and Ecological Verifications") is described as an example for educational software written in APL. The software was developed at the Department for Structural Analysis at the Technical University of Graz, and has been successfully used for teaching building-physics. A.T.O.N. was designed in only four months. It comes with an user-friendly interface, which is easy and effortless to learn. EvAn object-oriented building-physics software A.T.O.N. (which is the acronym for the German "General Thermal and Ecological Verifications") is described as an example for educational software written in APL. The software was developed at the Department for yPro(tm) as an interactive development environment.

APL Generated Teaching and Testing Items to Enhance a Student's Ability to Discover Functional Relationships, Alvin J. Surkan, Lincoln, NE. Ideas and designs for programs to generate teaching examples are introduced. Some of these ideas are tested and demonstrated by a prototype APL program. Descriptions of the internal structure and requirements of the example generators are presented. The examples produced by the program are designed to motivate students to discover rules or relationships capable of mapping the in-domain inputs to correct outputs. Experimental examples used for demonstrating typical concepts and mathematical processes include the functions: ⍴ , ⍳ , × , ⍟ , ÷ , | and ⍕ with their intrinsic APL definitions. Initially, the types of arguments for these examples will be kept simple at first to promote learning such mathematical functions by permitting only single scalar arguments. Later, for discovery learning, the functions may be selected to permit or require arguments that are vector or array objects. An important component in the design of this system is parameter specification. Parameters determine the number of examples and their perspicuity. A student using the system may request additional examples that are easy to comprehend before the function is discovered from the program generated examples provided. The example-generating program also facilitates checking that establishes that a student's response is operationally equivalent to a correct one.

GFSR Pseudo-random Number Generation Using APL, Charles Winton, Jacksonville, FL. This paper demonstrates the effectiveness of APL for implementing the generalized feedback shift register (GFSR) approach to producing a random number stream. Various approaches to generating streams of pseudorandom numbers computationally have been devised, dating at least as far back as Norbert Wiener. These are normally discussed in the context of a course in modeling and simulation, since there may be practical implications to consider when building simulation models in APL or any other high level language. This is particularly the case for those circumstances when a (pseudo) random number sequence needs to be repeated, or when multiple streams are needed. In such cases a pseudorandom number generator other than that supplied in the programming language command set needs to be implemented. The linear congruential method for pseudorandom number generation is still commonly used, and is easily implemented in APL or any other high level language; however, it is known to have undesirable *n*-space uniformity characteristics. Moreover, the period length of its random number stream is limited by the underlying machine's word size. This is a serious issue, since at present day computer speeds, a simulation run could exhaust such a random number stream in a few hours. Very long period (VLP) pseudorandom number generators remedy this deficiency.

One class of these, generalized feedback shift register (GFSR) pseudorandom number generators, is based on algebraic manipulation of irreducible trinomials of high order. The manner in which this is accomplished particularly lends itself to elegant APL implementation.

Tutorials and Workshops

Chaos and Symmetry Exhibit by Bruce Adcock, Ned Allis, Jeffrey P. Dumont, Flynn J. Heiss, Kevin C. Jones, Clifford A. Reiter, and Lisa M. Vislocky. The fact that chaos is compatible with symmetry is intriguing. This exhibit is a visual exploration of chaos with symmetry created with J. This includes explorations of planar crystallographic symmetries, their evolution from one symmetry type to another reminiscent of Escher's Metamorphosis, and explorations near forbidden symmetries. Chaos in space with 3-dimensional symmetry is also considered. J provides a remarkably flexible environment for the creation and exploration of these complex, yet structured, objects.

The Creation of Chaos with Symmetry. Bruce Adcock, Ned Allis, Jeffrey P. Dumont, Flynn J. Heiss, Kevin C. Jones, Clifford A. Reiter, and Lisa M. Vislocky. The creation of images of chaos containing symmetry will be described. This includes a discussion of the power of using J as an exploratory tool. Indeed, the generality of J makes it possible to create images of any of the 17 planar wallpaper groups or any of the 230 space groups using the same handful of adverbs and conjunctions. In addition to considering a wide variety of two and three dimensional symmetry groups, explorations near forbidden symmetry will be illustrated. Animations showing the evolution from one planar symmetry type to another will be shown in addition to rotations of three dimensional symmetric chaotic objects.

ISAP (Lingo Allegro's high level interface between Microsoft's Internet Information Server and Dyalog APL), Willi Hahn (whahn@fh-bingen.de)

Introduction - What it is all about, how to set up the environment
A very simple example - Introducing the concepts
German Lotto - See how it works, obviously APL is involved
One may calculate - Dealing with numerical variables
Creating an account - A common task in e-business
Send email - Without relying on the browser's mail client
A bookstore - More details on the joint paper I'm going to present with Bob Brown
APL with a WEB interface - How APL could be advertised over the Internet
Graphics - How to produce fast business graphics with ISAP on the Internet
Connect to a database - Showing how APL works as an integrating tool
Code browsing - Showing APL code on Web Pages
Errors to avoid - Report on some experience
Resume - What it is, what it is not

Window Models and Management, Paul Mansour, President, the Carlisle Group, Inc. Discusses MDI, SDI, workspaces, workbooks, projects and namespaces.

Reading the Dictionary of J, Martin Neitzel (neitzel@gaertner.de). This workshop is for those who found the *Dictionary of J* unapproachable or feel not at ease with it.

We will read just a few pre-selected sections but these with a great attention to detail. After attending this tutorial, you should be able to answer questions involving items, cells, and rank on your own with a bit more confidence.

J Programming in a Unix environment, Martin Neitzel. The demonstration/discussion workshop presents how J programs can be developed in and for the Unix environment. Specific topics will be:

- J in a multi-user environment
- interactive J: approaches to session management
- batch-oriented J runs
- communicating between J and the outside world: stdio, host commands and pipes, TCP/IP, shared library interface
- useful Unix tools for the J programmer
- using the juggle.gaertner.de information central

The overall purpose of the workshop is threefold:

- Introduce relevant aspects of the Unix environment to those J programmers usually working with Windows
- Exchange practical tips & tricks between users of J under Unix
- Discuss possible future directions for the J system on Unix

The J Workbench, Martin Neitzel. The J Workbench is a session manager geared towards tacit programming. It replaces the usual scrolling execution window of the Windows J system with a form of several input and output fields. The form separates test arguments from the verb under development. A history is kept of the test arguments and switching between verbs happens in a hypertext-like fashion without the need for explicitly coded references. The workbench allows debugging fragments of tacit phrases simply by highlighting the points of interest.

The tutorial focuses on the presentation of the J Workbench from a user's perspective. However, the J source of the Workbench is freely available and a quick tour through its organization will be given to ease the life of daring contributors.

Developing Quality Business and Statistical Graphics with Causeway RainPro, Alan Sykes, Swansea University. Workshop, with hands-on PCs. This workshop will cover the basic techniques of chart design using the *RainPro* tools from Causeway. All the experience gained will be equally applicable to APL+Win and Dyalog APL. Attendees will learn how to construct a variety of composite charts, and how this code can be assembled into an effective utility set for generating many standard statistical presentations, such as multiple box-whisker plots.

The workshop will also cover the various publication options now available, for example in the production of PDF documents or HTML pages using the latest capabilities of Dyalog 8.2. Delegates will be shown how to make 'active' graphics where clicking on particular data items might route a user to another page on a web site.

All the examples are illustrated on the Causeway web site — please check out www.causeway.co.uk/apl99/ before joining the workshop, and bring your own examples along too.

Using HTML for Forms Design in APL, Fred Wald (Fwaid@acm.org). Workshop: half-day, hands-on. This workshop will demonstrate the use of his HTML parser written in APL, and give the attendees experience in using HTML as a page-description language to design forms which are then used an APL application. The use of an existing WYSIWYG html editor to create the HTML, and hence the forms, will be covered. This allows a de-coupling of the APL coding and forms design processes. The attendees will also be exposed to creating HTML scripts dynamically, with APL code, and then using those created scripts to build forms. The ability to capture forms from the web, or other HTML source, and incorporate them into an APL application, will also be covered.

APL and the Internet, Peter Donnelly, Dyadic Systems, Ltd. In this workshop, students will learn how to implement an APL application as a Web SERVER, how to run APL in a Web BROWSER, and how to combine the two for a wholly APL web client/server solution. The workshop is based upon the APLSERVE software supplied with Dyalog APL Version 8.2.

Dynamic Functions "Masterclass", John Scholes, Dyadic Systems, Ltd. The workshop will discuss a range of techniques available to the Dynamic Function programmer. It will include programming in a functional style and will address performance issues.

APL and OLE, John Daintree, Dyadic Systems, Ltd. This workshop will show students how to drive applications such as Excel and Access, or components such as Active Data Objects (ADO), from Dyalog APL APL using COM/OLE interfaces. Mechanisms for accessing other interfaces such as those provided by the Window shell and DirectX will also be covered.

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, at 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 APL/W 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 APL/W for Windows	poa	Finnish distributor of Dyalog APL products.
	Dyalog APL for Unix	poa	See Dyadic's listing for product details.

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.
IAC/Human Interfaces			
	I-APL/Mac	13	Macintosh version of I-APL
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. 5604-260. 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 additionals $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	APL*PLUS/PC	poa	APL systems marketed and supported ...
	Dyalog APL	poa	from: Dyadic, Manugistics, IBM
	APL2	poa	under: Windows, OS2 and Unix
Iverson 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	Manuglstics Products and ISI	poa	New Zealand distributor
MicroAPL	APL.68000 Level I	2000	First generation APL with numerous enhancements. Multi-user version (Unix, Mirage, MCS).
	APL.68000 Level II	2500	Second generation APL. Nested arrays, user defined operators, selective specification etc. Multi-user version (Unix, Mirage, MCS)
	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 I Mac, ST, Amiga	87	First generation APL. Single user, full windowing interface, software floating point support.
	Mac, Amiga	260	First generation APL. Single user, full windowing interface, hardware floating point.
	APL.68000 Level II ST	170	Second generation APL. Full windowing interface, software floating point support.
	Amiga	260	Second generation APL. Full windowing interface.Hardware and software floating point support.
	Mac	520	Second generation APL. Full windowing interface.Hardware and software floating point support.
	APL*PLUS Rel 10 APL*PLUS II V 4.0	450 1395	
Oasis	Dyalog APL	poa	Dyadic Systems
	APL*PLUS	poa	Manuglstics
	APL.68000	poa	MicroAPL Ltd
	APL2	poa	IBM
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+Unk		
	APL+UNIX		
	APL*PLUS Sharefile		
Soliton Associates	SHARP APL for MVS	poa	for IBM MVS mainframes
	SHARP APL for Unix	poa	for IBM RS/6000 and Sun SPARC
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 Software	Adaytum Planning	poa	Full-featured Budgeting and Financial Planning system for medium to large enterprises.
APL Group (see Eventra)			
APL Software\Services			
	APL Utilities	poa	DOS-based APL software for .AWS, .SAW, .TRY, . IWS, APL Conferences, utilities, unlocking (.AWS), and more. Send request for an email catalog to 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 *NewLeaf* reports. Functionally compatible across Dyalog/W and APL+Win.
	NewLeaf for Dyalog and +Win	400	Frame-based reporting tool with comprehensive table-generation and text-flow support. Offers multiple master-page capability, bitmap wrap-around and on-screen preview with pan and zoom. Fully supported on Dyalog/W and APL+Win (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.
	DynaLink	poa	An ODBC client interface for DICE and IBM APL2 programs.
Eventra	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
IAC/Human Interfaces	SPARKS	poa	Educational simulation of electric circuit (for Apple Mac.)
	EPIDEMIC	poa	Educational simulation of spreading infection (for Apple Mac.)
	COINS	poa	Educational simulation (KS3) of coin-tossing experiment with simple stats (for Apple Mac.)
	FIBONA	poa	Educational simulation of Fibonacci's rabbits (for Apple Mac.)
I-APL Ltd	Educational workspaces	5	PC format disks with the examples from: Thomson, Espinasse (Kits 1-4), Kromberg, Jizba & FinnAPL. All the examples to save your fingers!
IBM APL Products	A Graphical Statistical System	$250	for DOS, Product Number 5764-009
	(AGSS)	$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	IUTILS/XP	20-95	Cross-platform utility library including simple OS calls (DIR, COPY, DEL, RENAME) and DATE functions. For APL*PLUS II, APL2 and Dyalog APL under Windows, OS/2 and Unix.
	ASI	95	APL Spreadsheet Interface. "Device-Independent" spreadsheet driver supporting Excel, 123 and Quattro-Pro for Dyalog APL/W
	WinCom	95	Asynchronous comms package for Dyalog APL/W
	S2D,22D,X2X	poa	Advanced APL syntax analysis and conversion packages from Sharp and APL2 to Dyalog, and between any two APLs
	SQAPL Client	poa	Interface from APL*PLUS II, APL2 and Dyalog (Windows, OS/2 or Unix) to most SQL databases over most networks.
	SQAPL Server	poa	Makes APL*PLUS II, APL2 or Dyalog APL (Unix) available as SequeLink servers. Can be called from SQAPL clients or other applications such as Excel, C++, Smalltalk, Visual Basic.
JAD Software	JAD SMS	poa	Software management system for Dyalog APL (7.2 or later) based on hierarchical databases; includes gui interface and stand-alone functions. Pricing will be per-user. Available fall 1999.
Lescasse Consulting	APL+Win Monthly Training Program	$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..
Soliton Associates	LOGOS	poa	Application Development Environment
	MAILBOX	poa	Electronic Mail
	VIEWPOINT	poa	Report generator with interfaces to DB2 and MVS data
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.
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
Adfea	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.
Dinosoft Oy	Consultancy	poa	Specialised in very large databases.
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	Consultant with 10+ years experience with various APL interpreters and C.
IAC/Human Interfaces	Consultancy	poa	APL and ergonomics consultancy services.
	Documentation	100-200	On-line assistance, product demos & mock-ups, manual writing; foreign language software localization.
	Training	poa	Using I-APL for courseware & distance learning materials; Mac programming in C, APL & HyperCard.
INFOSTROY	Consultancy	poa	Moving applications between platforms. Client/server development. Multilingual user interface.

Insight Systems	Consultancy	poa	Experts in APL conversions between any combination of: APL*PLUS, APL2, Dyalog APL and Sharp APL. We are also experienced right-sizers, comfortable with networks and relational databases (that also means when NOT to use SQL) and client/server development in APL, C and Visual Basic.
JAD Software	Consultancy	poa	Systems design and development, project management, technical manuals, financial and actuarial expertise in APL.
Phil Last	Consultancy	poa	APL consultancy, modelling and programming.
Lescasse Consulting	Consultancy	poa	A range of consultants, experts in Windows programming, with APL+Win and Dyalog APL/W. More than 100 major APL applications already developed. We all have additional expertise in Formula One and Delphi.
Lingo Allegro	Consultancy	poa	General APL consulting, Internet website development, migration and downsizing, performance tuning, education and training.
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, Y2K, 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.
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.
Stephen Wynn	Consultancy	poa	Most experience of financial planning, and mathematical areas: operational research, quality control, experimental design.

OTHER PRODUCTS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adfee	Employment	poa	Contractors and permanent employees
APL-385	Typefaces	poa	Variants of the APL2741 typeface available to specification.
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.
Soliton Associates	MVSLINK	poa	Interface from Sharp APL (Unix & MVS) to non-APL data and software in the MVS environment.
	SSQL	poa	High-performance DB2 interface for Sharp APL (Unix and MVS).

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 Qty 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 Software	Douglas Rowley	13 Great George Street, BRISTOL BS1 5RR, UK. Tel: 0117-921 5555
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
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, Siebenbrunnenfeldg. 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 Group (see Eventra)		
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: 106173.3347@compuserve.com
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	Roolnberg 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
Bloomsbury Software	Peter Day	Bloomsbury House, 74-77 Great Russell St., London WC1B 3DA, UK. Tel: 0171-436 9481 Fax: 0171-436 0524 Email: pd@bloomsbury-software.co.uk
Camacho	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS, UK. Tel: 0117-973 0036. email: 100612.1057@compuserve.com
Ray Cannon		21 Woodbridge Rd, Blackwater, Camberley, Surrey GU17 0BS, UK. Tel: 01252-874697 Email: 100430.740@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: 0171-404 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
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	Per Gjerløv	Email: gjerlov@ibm.net
Dinosoft Oy	Pertti Kalliojärvi	Lönnrotinkatu 21C, 00120 Helsinki, FINLAND. Tel: +358 9 70028820 Fax: +358 9 70028824 Email: dinosoft@dinosoft.fi
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)-782-6648 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	Bertellusvagen 12A, S-146 38 Tullinge, 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	Ken Chakahwata	114 Lemsford Lane, Welwyn Garden City, Herts AL8 6YP, UK
Analytics Ltd.		Tel/Fax: 01707-339620. Email: KenChakahwata@compuserve.com
General Software Ltd	M.E. Martin	Little Wester House, Westerhill Road, LINTON, Kent ME17 4BS Tel: 01622 749328 Fax: 01622 749365 E-mail: martin@gsoft.freeserve.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 Ltd		Hamilton House, 1 Temple Avenue, Victoria Embankment, London EC4Y 0HA, UK. Tel: 0171-353 8900; Fax: 0171-353 3325; Email:100020.2632@ compuserve.com
Hoekstra Systems Ltd	Bob Hoekstra	Dominique, Salisbury Road, Woking, Surrey, GU22 7UR, UK. Tel: 01483-771028 Email: bob.hoekstra@khamsin.demon.co.uk
Michael Hughes		28 Rushton Road, Wilbarston, Market Harborough, Leics. LE16 8QL, UK. Tel: 01536-770998 Email: Michael@Hughes.uk.com
IAC/Human Interfaces	Ian A. Clark	IAC Human Interfaces, Unit R21, Auckland New Business Centre, St. Helen Auckland, Bishop Auckland, County Durham, DL14 9TX, UK. Tel: 01388-605544. Email: 100021,3073@compuserve.com
I-APL Ltd	Anthony Camacho (for queries, order forms)	11 Auburn Road, Redland, Bristol BS6 6LS, UK. Tel: 0117-973 0036. Email: 100612.1057@compuserve.com
	J C Business Services (for pre-paid orders only)	56 The Crescent, Milton, Weston-super-Mare, Avon, BS22 8DU, UK. Tel: 01934-625181
IBM APL Products	Nancy Wheeler	APL Products, IBM Santa Teresa, Dept M46/D12, 555 Bailey Avenue, San Jose CA 95141, USA. Tel: +1 (408) 463-APL2 (=2752) Fax: +1 (408) 463-4488 Email: APL2@vnet.ibm.com Cserve: GO IBMAPL2
INFOSTROY	Alexei Miroshnikov	3 S. Tulenin Lane, St. Petersburg 191186 Russia. Tel:+7 812 312-2673 Fax:+7 812 311-2184 Email:alm@infostroy.spb.su
Insight Systems ApS	Morten Kromberg	Nordre Strandvej 119C, DK-3150 Hellebaek, Denmark Tel:+45 49 76 20 20 Fax: +45 49 76 20 30 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
Japan APL Assoc	Toshio Nishikawa	1-8-13 Masujima Build.6F Higashi Gotanda Shinagawa-ku, Tokyo Japan 141-0022. Tel: +81 (03) 3280-0411 Fax: +81 (03) 3280-0418 Email: KYY00361@niftyserve.or.jp
J Austria	Joachim Hoffmann	Bramley, Oldstead, YORK YO61 4BL. Tel: 01347-868121. Email: JoHo@ping.at
Phil Last Ltd	Phil Last	146 Crossbrook Street, Cheshunt, Herts, EN8 8JY, UK. Tel: 01992-633807 Fax: 0121-359 0375 Email: phil_last@compuserve.com
Lescasse Consulting	Eric Lescasse	18 rue de la Belle Feuille, 92100 Boulogne, France. Tel: +33.1.46.05.10.76 Fax: +33.1.46.04.60.23 Email: eric@lescasse.com
Lingo Allegro USA, Inc	Steven J Halasz	1105 Chicago Avenue, Suite 155, Oak Park, IL 60302, USA. Tel:+1 800 546 4621-1 Email: sjhalasz@interaccess.com
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: 01442-834015 Email=macleod@entropy-group.freeserve.co.uk
Mercia Software Ltd.	Gareth Brentnall	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	South Bank Technopark, 90 London Road, LONDON SE1 6LN, UK. Tel: 0171-922 8866 Fax: 0171-928 1006 Email: MicroAPL@microapl.demon.co.uk
Milinta Inc.	Dan Baronet	4215 St-Andre, Montreal, CANADA H2J 2Z3. Tel: +1 (514) 529-1434
Ellis Morgan	Ellis Morgan	Myrtle Farm, Winchester Road, Stroud, Petersfield, Hants GU32 3PE, UK. Tel: 01730-263843 Email: Ellis@mrtlfrm.demon.co.uk
NY Sig	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
Optima Systems Ltd	Paul Grosvenor	115 Brighton Road, Purley, Surrey CR8 4HE, UK. Tel: 0181-763 2490 Fax: 0181-763 2491
RadSys Technologies AB	Randolph Schrab	Lovsangarv. 18, S-756 52 Uppsala, Sweden. Tel: +46 18 32 41 53 Fax: +46 708 1996 11 Email: randolph.schrab@radsys.se
Renaissance Data Systems	Ed Shaw	P.O. Box 313, Newtown, CT 06470, USA. Tel: +1 (203) 270-9729 Email: rendata@apibooks.cnchost.com or orders@apibooks.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	Casella 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: seapldoc@aol.com
Shepp & Associates LLC	Andrew Shepp	1312 Washington Avenue, 6th Floor St. Louis MO 63103, USA Tel: +1 (314) 621-3272 Fax: +1 (314) 621-4257 UK Address: Claridge House, 29 Barnes High St, London SW13 9LW Tel: 0181 8768666 Fax: 0181 8768660 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@eecg.toronto.edu
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
Sollton Associates	Laurie Howard	Sollton 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-5, 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: amfaust@aol.com
Rex Swain	Rex Swain	8 South Street, Washington, CT 06793 USA. Tel: +1 (860) 868-0131 Fax: +1 (860) 868-9970 Email: rex@rexswain.com
SwedAPL	Christer Ulfhielm	Novator Consulting Group AB, Svärdvägen 11C, S-182 33 Danderyd Sweden. Tel: +46 8 622 63 50 Fax: +46 8 622 63 51 CServe: 100341,404
Swiss APL User Group		Swiss APL User Group, CH-3001, Bern 1, Switzerland Email: sl@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
Stephen Wynn		8 Clarence Gardens, Brighton, Sussex BN1 2EG, UK. Tel: 01273-327238 Email: centre@mistral.co.uk
Zark Incorporated	Gary A. Bergquist	23 Ketchbrook Lane, Ellington CT 06029, USA. Tel: +1 (860) 872-7806

WORLD WIDE WEB SITES

ACM SigAPL	www.acm.org/sigapl/
Adapta Software	www.adapta.de/
Adaytum Software	www.adaytum.com/
AFAPL	www.ensmp.fr/~scherer/langlet/ (Journal available on line)
APL2000	www.APL2000.com/
APL-385	www.demon.co.uk/apl385/
AUSCAN	www.interlog.com/~rjp/auscan/
Eke van Batenburg	wwwbio.LeidenUniv.nl/~Batenburg/index.html
Capital PC User Group	http://cpcug.org/
Causeway	www.causeway.co.uk/
CODEWORK	www.inrete.it/cdwk/eng/homee.html
COSY (Bob Armstrong)	www.cosy.com/
Dinosoft Oy	www.dinosoft.fi/
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.torolab.ibm.com/ap/apl/apl2.html
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/
Lescasse Consulting	www.lescasse.com/
Lingo Allegro USA Inc	www.lingo.com/
Mackay Kinloch	ourworld.compuserve.com/homepages/Alastair_Kinloch/akinloch.htm
MicroAPL Ltd	www.microapl.co.uk/
Oasis b.v.	www.oasis.nl/
Open Directory	http://dmoz.org/Computers/Programming/Languages/APL/
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/
Strand Software Inc.	www.jsoftware.com/
Rex Swain	www.rexswain.com/
Toronto SIG	www.torontoapl.org/
Jim Weigang	www.chilton.com/~jimw/

FTP SITES

IBM APL2	ps.boulder.ibm.com/ps/products/apl2/
Toronto toolkit	see Toronto SIG home page
Waterloo Archive	archive.uwaterloo.ca/ftparch/languages/apl
APL-to-ASCII	archive.uwaterloo.ca/languages/apl/workspaces/aplascii

Microsoft Announce New APL

http://idt.net/~alstone/fsfsapl.htm#apl_prog_env

LANGUAGE FEATURES/LIMITATIONS

Language features

- BASIC like syntax.
- A command can start anywhere on a line, so indention can easily be applied.
- A space or blank characters can be applied anywhere within the program code. When a space is required by program syntax any number of spaces are allowed (the total number of characters on the line must not exceed 80).
- The Language syntax is case in-sensitive. The variables "TEMP" and "temp" are the same. The compiler converts everything to uppercase anyway (except for string constants). So those who want to employ fancy stuff like hungarian notation, etc. the compiler won't care.
- Support of limited Keyboard interaction.

Additional APLC32 features

- Support of functions and procedures
- Local variables
- String variables
- Preprocessor directives (include, pragma)
- 64K file size limit does not apply
- Detection of aplc32 compiled adventures
- Line size limit up to 2048 with lexical item limit of 512

Language limitations

- One command or statement allowed per line
- No support of string variables
- No support of File I/O
- Number range only between -32768, and +32767
- Very little support of flight control manipulation
- 64K binary adventure size limit

The APL Programming Environment

APL programs can be written in any standard ASCII text file editor. The APLC compiler will compile text files into binary form for FS5. The compiler can be executed from the DOS command prompt. For use with FS6/98 they must either be compiled with the APLC.EXE into .ADV format and converted using the Microsoft (henceforth referred to as MS) supplied converters FSCONV.EXE [1] and FSCONV98.EXE [2] to FS6/98 format or compiled using the Freeware APLC32 compiler [3] (henceforth referred to as APLC32).

```
Syntax:
        APLC   "Input file" ["Output File"]
```

The first argument after the program name is the name of the APL source file (APLC assumes a ".TXT" file extension, so that is optional). The second argument is the name of the adventure binary file (any extension given is stripped, and ".ADV" is used).

If errors in syntax are encountered during compilation they will be displayed on the screen or console window complete with line numbers. The line number pointing to the location of the error includes blank lines. So it would be that line in your editor, no need to worry about the count being off due to blank lines, they are included in the count.

APLC32 contains extra command line options which can be used at compile time.

-c	Check syntax only, do not write the output file
-k	Attempt to convert old source file as it is compiled
-o	Disable extra optimisations
-oo	Disable all optimisations
-s	Send error output to stdout rather than stderr
-w	Do not produce warnings during compilation
-v	Turn off assignment checks for system variables
-a	Turn off altitude compensation
-fy or -fn	Turn on or off output directory forcing
-x	Turn off any extensions, useful to check compatibility
-95	Generate output for FS6 (ver 1.18+).
-98	Generate output for FS98 (ver 1.18+)

See the APLC32.DOC file in [3] for more details of these command line options.

The Basic Structure of FSFS APL Programs

APL is a programming language similar to the old DOS GW BASIC Language in format and syntax.

```
Example:

 TITLE "CMI to ORD adventure"
 DESCRIPTION "Champaign to Chicago adventure"
 ;Test adventure
 ;
 DECLARE "variable names"
 ....
 ....  (program code)
 END
```

Unlike languages like C or PASCAL no particular keyword or value is required (like main, start, begin, *etc.*) to successfully compile and execute an adventure. Though none of the keywords listed in the above example are required, we shall consider this the standard format of what should be in every adventure.

For more information please refer to http://idt.net/~alstone/fsfsapl.htm

References

[1] http://www.microsoft.com/games/fsim/converter.htm

[2] ftp://ftp.microsoft.com/deskapps/games/public/flightsim/ fsconv98.exe

[3] ftp://ftp.iup.edu/flight-sim/uploads/aplc119.zip

Zark Newsletter Extracts

introduced by Jon Sandles

Here we are again: another crossword and the solution to the Limbering Up column from last month. After a deluge of comments to the relatively simple problem set in Vector 15.2 and reported in Vector 15.4, we have had no further comments (but see Anne Wilson's letter on p.10 which arrived after I had written this). Once again, I encourage you all to try and shed new light on these old problems.

Utility Corner: Tree Diagrams

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

In the last issue's LIMBERING UP column, we asked you to write a function named *TREE* that draws tree diagrams. The *TREE* function assumes the existence of three global variables: *<names>*, *<relations>*, and *<shown>*. *<names>* is a character matrix of the names of the entries in the tree. *<relations>* is a two-column matrix of indices into *<names>*, that describes the relations between the entries; the first column is the index of the "parents", the second column is the index of the "children". *<shown>* is a bit vector with one element per row of *<names>*, telling which entries have already had their children shown. *<shown>* is both used and updated by *TREE*.

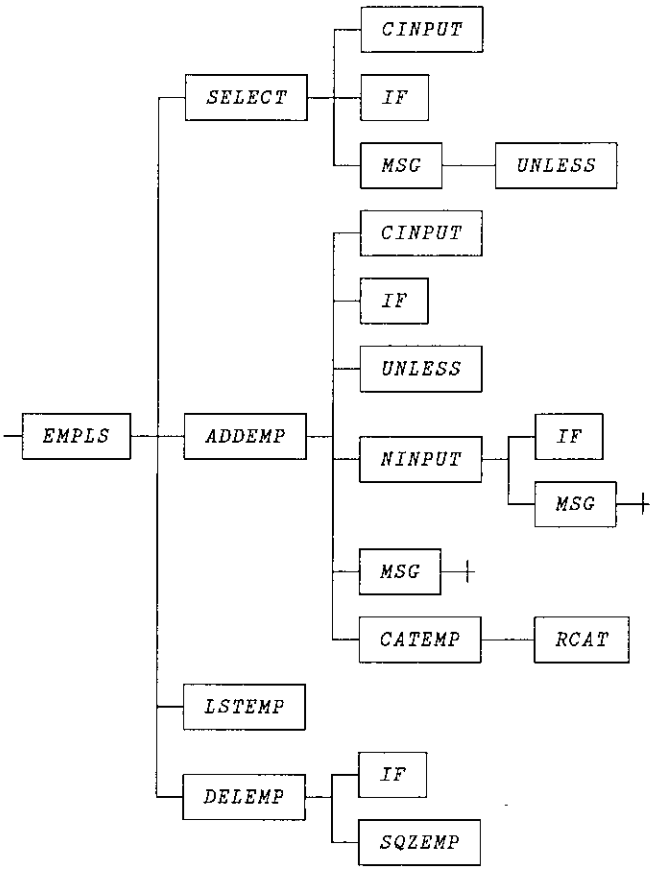
TREE must be recursive. That is, it must call itself.

Here are some examples of values for the three global variables:

```
      names
EMPLS
SELECT
CINPUT
IF
MSG
UNLESS
ADDEMP
NINPUT
CATEMP
RCAT
DELEMP
SQZEMP
LSTEMP
      shown
0 0 0 0 0 0 0 0 0 0 0 0 0 0
      relations
 1  2
 2  3
 2  4
 2  5
 5  6
 1  7
 7  3
 8  4
 8  5
 7  4
 7  6
 7  8
 7  5
 7  9
 9 10
 1 13
11  4
11 12
 1 11
```

The right argument of *TREE* is the index (into <*names*> and <*shown*>) of the entry to be displayed, along with its children, and their children, and so on. The result of *TREE* is a character matrix.

Given the above global variables, here's an illustration of the use of *TREE*:



(This diagram is a bit different from the one presented in the last issue of Zark APL Tutor News. In particular, extra blanks have been squeezed from short names, and each entry and its corresponding children occupy a rectangular block of space. These changes simplify the *TREE* function and improve the appearance of its output.)

The need (or convenience) for *TREE* to be recursive is apparent when you consider that the result of *TREE* `1` contains the results of *TREE* `2`, *TREE* `7`,

TREE 13, and *TREE* 11. Likewise, the result of *TREE* 2 contains the results of *TREE* 3, *TREE* 4, and *TREE* 5.

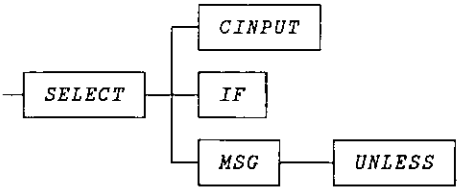
Here are some examples to illustrate the point:

```
      shown[]←0     ⍝ Reset to all zeros
      TREE 3
```


```
      TREE 4
```

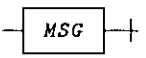

```
      TREE 5
```


```
      shown[]←0     ⍝ Reset again
      TREE 2        ⍝ Includes 3, 4, and 5
```

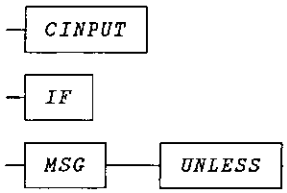


The overall logic of the *TREE* function is fairly simple. If asked to display an entry that has no children, return the name in a box, squeezing out extra blanks:

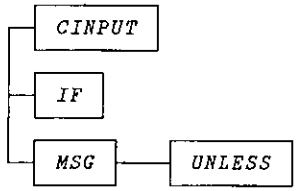
If asked to display an entry that has children, but the children have already been displayed elsewhere (as indicated in <*shown*>), include a + after the box:



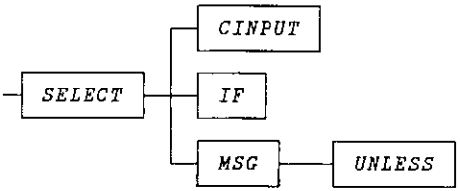
If asked to display an entry that has children, and the children have not yet been displayed, call *TREE* for each of the children, and catenate the results row-wise, padding as necessary so they each have the same number of columns:



Then, catenate a line along the left side:



Finally, put the name of the parent entry in a box, pad it with enough blank rows to make it the same size as the catenated children (centred), and stick it on the left side:



Before we list and discuss the *TREE* function, we need to consider design philosophy. We received a number of rebukes from respondents for our use of global variables.

Typical was the comment from Charles Haspel:

> In my code, and that of anyone I teach, globals are forbidden. Any function ought to be just that, a function of its arguments without side effects. This practice makes it easier to describe and prove functions, it allows them to be tested without affecting the workspace they are in, it allows more than one copy of a function to be run at a time and it makes changes much easier by containing the universe affected by the change... I hope I can convince you to eschew globals for your valuable teaching articles in the Tutor News.

We're convinced. Global variables should be avoided when possible. Global variables require documentation in both the calling function and in the subfunction. For example:

```
      ∇ MAINFN;VAR
[1]     VAR←whatever
[2]     SUBFN ⍝ Requires VAR
      ∇
```

```
      ∇ SUBFN
[1]     ⍝ Requires VAR
[2]     Z←⍴VAR
      ∇
```

In this case, the variable *VAR* is global to *SUBFN*. The comment in *MAINFN* is required so the programmer remembers not to change the name of *VAR* without changing it in *SUBFN* as well. The comment in *SUBFN* is required so the programmer knows where *VAR* came from, and that it is intentionally global, not accidentally unlocalised. If either comment is omitted, the code is confusing.

Fortunately, the need to comment globals provides motivation to find an alternative. For example, the use of arguments eliminates the need for global variables and their documentation:

```
      ∇ MAINFN;VAR
[1]     VAR←whatever
[2]     SUBFN VAR
      ∇
```

```
      ∇ SUBFN VAR
[1]     Z←⍴VAR
      ∇
```

Whenever possible, global variables should be avoided!

Having said this to ease our guilt, we're still going to pass global variables to the *TREE* function. After all (we rationalise), *TREE* is not our everyday function. It's recursive. Working with recursive functions is like working in another universe. Different rules apply.

For example, a recursive function can generate a deep state indicator (by recursing). If the values required by the function are passed as arguments rather than globals, you'll very quickly generate numerous copies of the same arrays.

From a storage viewpoint, this is inefficient and may generate as *WS FULL* message.

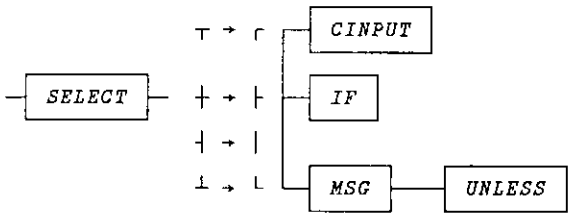
So, let's use global variables, but which we didn't have to, and let's be fastidious about documenting them, and be careful about limiting their side effects.

Below is one version of the *TREE* function. This one works in APL*PLUS PC or APL*PLUS II. The ⋄ symbol (diamond) is a statement separator to allow more than one statement on a line. The ⍪ symbol (cat-bar) does first-dimension catenation (, [1]). The line-drawing characters are those available on the PC.

```
     ∇ CMAT←TREE INDEX;C;D;E;J;K;R;S
[1]    ⍝ Returns a character matrix tree diagram for
[2]    ⍝ the tree organisation described in the
[3]    ⍝ globals:  names, relations, shown.  <names>
[4]    ⍝ is a character matrix of the names of
[5]    ⍝ entities in the tree.  <relations> is a 2-
[6]    ⍝ column matrix of indices into <names>, that
[7]    ⍝ describes the relations between the
[8]    ⍝ entities; the first column is the index of
[9]    ⍝ the parent, the second column the index of
[10]   ⍝ the child.  <shown> is a bit vector with
[11]   ⍝ one element per row of <names>, telling
[12]   ⍝ which entities have already had their
[13]   ⍝ details (children) shown.  <shown< is both
[14]   ⍝ used and updated by TREE.  INDEX is the
[15]   ⍝ index into <names> and <shown> of the top
[16]   ⍝ level of the tree.  TREE is recursive.
[17]   ⍝ Its output looks like:
[18]   ⍝
[19]   ⍝
[20]   ⍝                    ┌─THOMAS─┐
[21]   ⍝                    │        │
[22]   ⍝                    │        │
[23]   ⍝    ┌─FRED─┐        ├─BOB────┤
[24]   ⍝    │      │        │        │
[25]   ⍝    │      │        │        │
[26]   ⍝    │      │        └─WILLIAM─┴─SID─┘
[27]   ⍝    │      │                 │     │
[28]   ⍝    └──────┘                 └─────┘
[29]   ⍝ Inds of children (relations below):
[30]    K←(INDEX=relations[;⎕IO])/⍳1↑⍴relations
[31]    CMAT←names[INDEX;] ⍝ Current name
[32]   ⍝ Del trlng blanks; incl one ldg/trlg blank:
[33]    CMAT←' ',(1++/∨\' '≠⌽CMAT)↑CMAT
[34]   ⍝ Enclose in box
[35]    CMAT←' ─ ','┌┤└','─'⍪CMAT,[⎕IO-0.5]'─'
[36]    CMAT←CMAT,'┐','│','├'[⎕IO+×⍴K],'┘'
[37]   ⍝ Have details been shown? flag them:
```

```
[38]   S←shown[INDEX] ⋄ shown[INDEX]←1
[39]   →(⍴K)↓0 ⍝ Exit if no children
[40]   CMAT←CMAT,' ─ ' ⍝ Put line from right centre
[41]  ⍝ Put cross to right and exit if shown:
[42]   →S↓L1 ⋄ CMAT←CMAT,' ┼ ' ⋄ →0
[43]  ⍝ Get first child recursively
[44]  L1:C←TREE relations[K[⎕IO];1+⎕IO]
[45]   →(1≠⍴K)⍴L2 ⍝ Branch if > one child
[46]   CMAT←CMAT,' ─ ' ⍝ Straight line for 1 child
[47]   →(3≠1↑⍴C)/L4 ⍝ Branch if parent needs padding
[48]   →L5
[49]  ⍝
[50]  ⍝ Loop by child:
[51]  L2:→(⍴K←1↓K)↓L3
[52]   D←TREE relations[K[⎕IO];1+⎕IO] ⍝ Next child
[53]  ⍝ Caten. children. padding to same no. cols:
[54]   R←(1↑⍴C)⌈1↑⍴D
[55]   C←(((1↑⍴C),R)↑C),⍪((1↑⍴D),R)↑D ⋄ →L2
[56]  ⍝
[57]  ⍝ Indices of line to left of each child:
[58]  L3:J←(C[;⎕IO]='─')/⍳1↑⍴C
[59]  ⍝ Insert line draw chars along left side:
[60]   R←⌊/J ⋄ S←⌈/J ⋄ C←' ',C
[61]   C[R+⍳S-R;⎕IO]←'│' ⋄ C[R;⎕IO]←'┌'
[62]   C[S;⎕IO]←'└' ⋄ C[1↓¯1↓J;⎕IO]←'├'
[63]   J←(⌈(1↑⍴C)÷2)--⎕IO ⍝ Centre ind of children
[64]  ⍝ Insert centre symbol
[65]   C[J;⎕IO]←(C[J;⎕IO]='│├┌┴')/'┤┼┬┴'
[66]  ⍝
[67]  ⍝ Incl blan rows on parent to centre:
[68]  L4:R←1↑⍴C ⋄ S←1↓⍴CMAT
[69]   CMAT←(R,S)↑((⌊(R-3)÷2),S)⍴' ')⍪CMAT
[70]  L5:CMAT←CMAT,C ⍝ Stick child to right
      ∇
```

The logic in the function is fairly straight-forward. One line that may seem confusing at first is [65]. On that line, the appropriate line-drawing symbol is chosen depending on where the parent unites with the line connecting the children. Perhaps this diagram will help you understand the logic:



A second version of the *TREE* function, shown below, works on mainframe APL2. Different characters are used in place of the PC's line-drawing characters. Also, nested arrays are used where convenient.

```
      ∇ CMAT←TREE2 INDEX;C;D;E;J;K;R;S
[1]    ⍝ Returns a character matrix tree diagram for
[2]    ⍝ the tree organisation described in the
[3]    ⍝ globals:  names, relations, shown.  <names>
[4]    ⍝ is a character matrix of the names of
[5]    ⍝ entities in the tree.  <relations> is a 2-
[6]    ⍝ column matrix of indices into <names>, that
[7]    ⍝ describes the relations between the
[8]    ⍝ entities; the first column is the index of
[9]    ⍝ the parent, the second column the index of
[10]   ⍝ the child.  <shown> is a bit vector with
[11]   ⍝ one element per row of <names>, telling
[12]   ⍝ which entities have already had their
[13]   ⍝ details (children) shown.  <shown< is both
[14]   ⍝ used and updated by TREE2.  INDEX is the
[15]   ⍝ index into <names> and <shown> of the top
[16]   ⍝ level of the tree.  TREE2 is recursive.
[17]   ⍝ Its output looks like:
[18]   ⍝
[19]   ⍝                  .---------.
[20]   ⍝               .-| THOMAS |-+
[21]   ⍝               | '---------'
[22]   ⍝     .-------. | .-----.
[23]   ⍝   -| FRED |-+-| BOB |
[24]   ⍝     '-------' | '-----'
[25]   ⍝               | .----------.    .------.
[26]   ⍝               '-| WILLIAM |---| SID |
[27]   ⍝                 '----------'    '------'
[28]   ⍝
[29]   ⍝ Inds of children (relations below):
[30]    K←(INDEX=relations[;⎕IO])/⍳↑⍴relations
[31]    CMAT←names[INDEX;] ⍝ Current name
[32]   ⍝ Del trlng blanks; incl one ldg/trlg blank:
[33]    CMAT←' ',(1++/∨\' '≠⌽CMAT)↑CMAT
[34]   ⍝ Enclose in box
[35]    CMAT←' - ','.|''',⍪'-'⍪CMAT,[⎕IO-0.5]'-'
[36]    CMAT←CMAT,'.','||'[⎕IO+×⍴K],''''
[37]   ⍝ Have details been shown? flag them:
[38]    S←shown[INDEX]
[39]    shown[INDEX]←1
[40]    →(⍴K)↓0 ⍝ Exit if no children
[41]    CMAT←CMAT,' - ' ⍝ Put line from right centre
[42]   ⍝ Put cross to right and exit if shown:
[43]    →S↓L1
[44]    CMAT←CMAT,' + '
[45]    →0
```

```
[46]  ⍝ Format all children at once:
[47]  L1:C←TREE2¨relations[K;1+⎕IO]
[48]   →(1≠⍴K)⍴L2 ⍝ Branch if > one child
[49]   CMAT←CMAT,' - ' ⍝ Straight line for 1 child
[50]   ⍝ Disclose and branch if parent needs padding:
[51]   →(3≠↑⍴C←↑C)/L3
[52]   →L4
[53]  ⍝
[54]  ⍝ Caten. children together, row-wise:
[55]  L2:C←⊃↑,/⊂[⎕IO+1]¨C
[56]  ⍝
[57]  ⍝ Indices of line to left of each child:
[58]   J←(C[;⎕IO]='-')/⍳↑⍴C
[59]  ⍝ Insert line draw chars along left side:
[60]   R←⌊/J
[61]   S←⌈/J
[62]   C←' ',C
[63]   C[R+⍳S-R;⎕IO]←'|'
[64]   C[R;⎕IO]←'.'
[65]   C[S;⎕IO]←''''
[66]   C[1↓¯1↓J;⎕IO]←'|'
[67]   J←(⌈(↑⍴C)÷2)--⎕IO ⍝ Centre ind of children
[68]  ⍝ Insert centre symbol
[69]   C[(C[J;⎕IO+0 1]∧.='|-')⍴J;⎕IO]←'+'
[70]  ⍝
[71]  ⍝ Incl blan rows on parent to centre:
[72]  L3:R←↑⍴C
[73]   CMAT←(((⌊(R-3)÷2),1↓⍴CMAT)⍴' '),[⎕IO]CMAT
[74]   CMAT←R↑[⎕IO]CMAT
[75]  L4:CMAT←CMAT,C ⍝ Stick child to right
      ∇
```

On [47], the each operator ¨ is used to format all children "at once." The result is a nested vector of character matrices.

On [55], the vector of character matrices is transformed to a vector of vectors of rows, then to a vector of all rows, and finally to a simple character matrix in which the rows of the child character matrices have all been padded and catenated (due to the conforming nature of disclose ⊃).

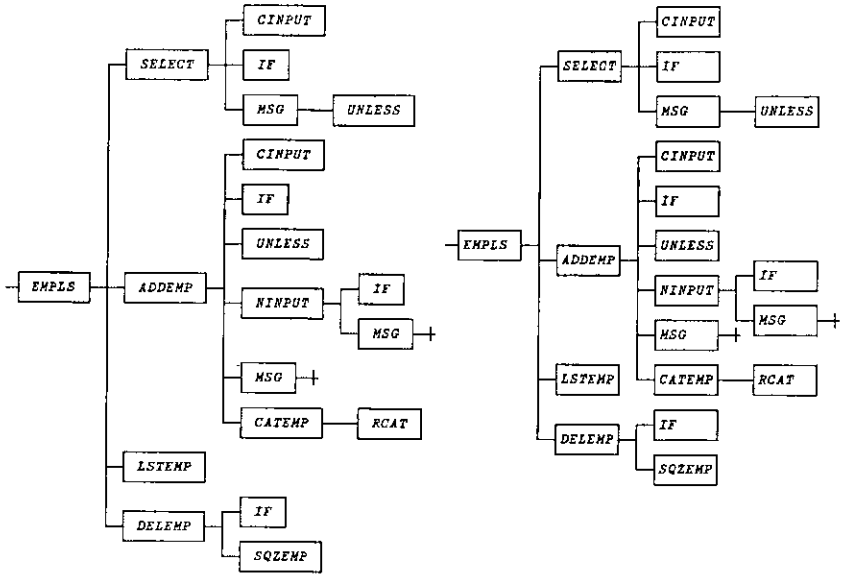
The other APL2 features used in *TREE2* are "first" (↑⍴*C* is like 1↑⍴*C* but returns a scalar) and "take with axis" (*R*↑[⎕*IO*]*CMAT* is like (*R*,1↓⍴*CMAT*)↑*CMAT*).

Our thanks to Charles Haspel for the APL2 techniques.

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

Comment by Jonathan Barman

I am feeling quite cross about the above solution for the tree problem. I have been helping out Vector production by typing up the Zark articles. When I typed up the previous Zark issue I did not have this issue, so I had to create my version of the *TREE* function so that I could include a picture of the result. My first version of *TREE* was very similar to the one above and took about ½ hour to complete. I then noticed that blanks had been removed so that the boxes were placed as close together as possible. The following shows the differences between the solution in this issue and the example result in the previous issue:



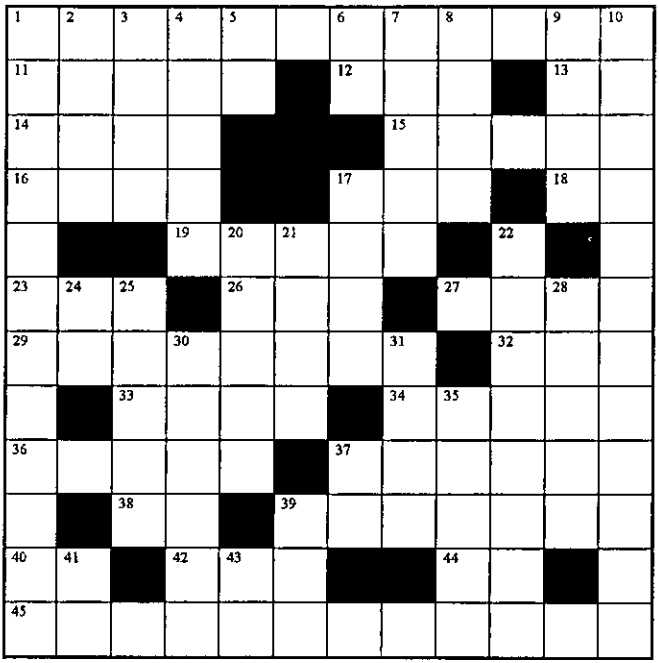
I found solving the blank removal problem really tough and it took me several hours to complete. I was quite looking forward to seeing how Zark had solved the problem, and was really annoyed to find that they had not provided a solution which exactly matched the original challenge.

Comment by Jon Sandles: I can only apologise to Jonathan for not passing on both articles at the same time! Still, it obviously gave him a good challenge! More seriously, I would like to thank Jonathan for typing these articles up, I am sure you all agree he is doing a great job. The question for Zark is: How did they produce those trees in the original article? Did they really cheat and not bother solving the problem until the next issue, and did they really then adjust the problem to make the solution easier?

Solution to Crossword from Vector 15.4

1 V	2 ×	3 ϕ	4 B	■	5 |	6 (	7 ?	8 9	9)	-	10 5
11 ι	ρ	V	,	12 B	■	13 M	5	ϕ	F	■	0
14 ⌈	A	-	.	5	■	15 ∧	/	V	N	>	0
/	■	16 1	0	2	17 -	.	2	×	S	■	↑
18 V	19 ~	2	5	■	20 N	=	5	2	■	21 →	0
■	A	■	22 ×	23 S	-	V	■	24 ÷	)	(	⌈
25 (	∨	/	V	∈	M	)	26 ι	1	■	27 ρ	K
28 ι	L	■	29 R	V	-	ι	1	0	+	V	C
30 0	↑	31 S	■	■	32 -	1	?	9	■	)	■
33)	P	C	O	34 P	Y	■	5	■	35 N	↓	V
36 ρ	ρ	A	■	I	■	37 ?	5	38 2	■	L	■
39 V	I	N	C	E	N	T	■	40 5	?	5	2

Crossword



Across

1. Are all elements of the vector A negative or greater than 10?
11. The running balance, given the starting balance M and the vector S of deposits and withdrawals.
12. Expand the matrix B by repeating each column six times
13. Any
14. All positive
15. $B \equiv V$ for vectors B and V
16. Half the length of
17. $-2 = S$
18. B isn't
19. The value of A in one year at annual interest rate K (e.g. 0.07)
23. $V * 2$
26. The number of dimensions of A
27. The natural log of M, but avoid a negative result or an error message

29. How many variable, function or label names have been used?
32. `3ρ'ρ',⍕2 2ρ'/\+↑'`
33. Are any of Z false?
34. `(20ρ0 0 1 0)/'(MATX+Z)*-(TAB÷K)×¯3'`
36. Does the vector B contain any nonzeros? (Compare to 20 Down)
37. `1÷V9×2`
38. A good variable name for a matrix of whole numbers
39. Extract the rows from matrix M9 corresponding to 1s in bit vector E9, and precede each row by the scalar N
40. The length of the vector X
42. `2⍕¯3↑7ρ'ρ/\)('`
44. The magnitude of M
45. Flag the rows of the matrix M that do not match the vector V (padded with blanks or truncated to conform with M)

Down

1. The indices in the matrix M of the rows that match the vector V (but omit the ninth symbol from the expression)
2. The first step to round A
3. `1⍕⍕'>+<×'`
4. 0 if A is numeric, blank if character.
5. What `)SAVE`, `)SI`, and `)SYMBOLS` have in common
6. `÷5÷3`
7. Are any of the elements of the vector B not the constant K?
8. The product of Booleans AB and S
9. `0=B-11`
10. Does the sorted vector V, which does not contain ¯99, contain duplicates?
17. The integer AL, incremented twice
20. Does the vector B contain any nonzeros (Compare to 36 Across)
21. The first element in the array OZ
22. The denominator, with redundant parentheses, of the expression that scales the values of the matrix M so each column adds to 1 (given `SM←ρM`).
24. S>0, for nonnegative S
25. Return bit vector BI if VY=0, or all 1s if VY=1
28. The largest reciprocal in the vector VM
30. `6ρ⍕'(A≠B)/M'~'(AB)'`
31. What to do often when power failures are common
35. The remainder of `M÷Z99`
37. `'÷×+-'[?⎕IO],,',`
39. N/1 for scalar N
41. The ____cist eliminates APL demons
43. The square root of one-third the sum of A and B (The first two symbols)

J-ottings 21: Time for amendment of data ...

by Norman Thomson

Item and atom are two words which differ only in a pair of vowels, yet in J their meanings are a world apart. What is an item? If you can't answer this with immediate assurance, then write no more J until you have read this article! Think of the USA as an object to be subdivided down to an eventual level of its individual citizens as *atoms*. If the highest level of breakdown is states, a typical *item* is Florida or California or Alaska. Wrap the first layer off the USA parcel, and ask how many items are inside. Answer, 50. Each of these items is itself an object and contains items, counties perhaps, and so the hierarchy continues. Conceptually the terms "object" and "item" cascade downwards, the former being always just one level higher in the hierarchy. Thus the object `a=: i.2 3 4`, has two items each of which is a 3 by 4 matrix. The "parcel unwrapping" verb which drops a level from object to item and reveals the shape of what is inside is `$"_1 :`

```
   $"_1 a                NB.  List the shapes of object a's items,
3 4                      NB.  namely two shape vectors
3 4
```

Introducing "amend"

Selective assignment as in `A[3 5 9]←14 25 90` in APL is a great convenience, but disguises the fact that there are really two processes telescoped into one. The first process involves a data transformation which selects those parts which are to be changed, while the second process does the actual replacement with a second set of data. Informally you can think of the selector as "sandwiched" between the new and old data as in the following illustration in which `a=: i.2 3 4` (that is 2 planes, 3 rows and 4 columns):

```
   new   selector  }  old

    (9)     1      }    a  NB.  Replace the second item (plane) with 9's
0   1   2   3               NB.  Parens are needed round either the 1 or
4   5   6   7               NB.  the 9 or both.   "Amend" ({) is
8   9  10  11               NB.  an adverb which qualifies "selector".
                            NB.  "Selector" is in general a verb, however
9   9   9   9               NB.  it can also be a noun as here.  Think
9   9   9   9               NB.  therefore of  1}  as a verb with
9   9   9   9               NB.  arguments "new" and "old".
```

The phrase `(9) 1} a` demonstrates that a J adverb, unlike an adverb in ordinary grammar, may qualify either a noun or a verb. A later example will show `}` qualified by a verb.

By default selection takes place at the level of items within objects, that is at rank 3, but the rank conjunction allows indexing to apply at lower levels:

```
12  9  14  15
16  9  18  19
20  9  22  23

   (9)  1}"2 a        NB.  Replace all second ROWS with 9's.
 0   1    2    3      NB.  Selection is made within rank 2 objects,
 9   9    9    9      NB.  that is vectors of vectors.
 8   9   10   11      NB.  These objects are the items of a.

12  13  14  15
 9   9   9   9
20  21  22  23        NB.  (9)  0}"0 a  replaces all atoms in a with 9.
```

A Data Amendment Problem

The problem is to masculinise American state names, for which two distinct kinds of technique are available, namely *transformation* followed by assignment, and *amendment*. With the former, and where "real" data is concerned, there is frequently a choice of two further possibilities, namely using simple lists or using boxed lists. Using lower and upper case names respectively to correspond to these options define

```
  USA=:'Florida' ; 'California' ; 'Alaska' NB.  Data definitions
  ]usa=:>USA
Florida                                    NB.  USA is a boxed list
California                                 NB.  usa is a simple list
Alaska

  masc=: ,&'o'@}:                          NB.  verb "masculinise" (simple)
  MASC=:masc each=:&.>                     NB.  verb "masculinise"  (boxed)
  MASC  USA                                NB.  masculinise a boxed list
```

Florido	Californio	Alasko

```
   dtb=:(.~i.&' ')                    NB.  delete trailing blanks (hook)
   (masc@dtb)"1 usa                   NB.  masculinise items of simple list
Florido
Californio
Alasko
```

Here are a few comments as asides:

(a) Compared with the boxed list the simple list incurs a space penalty imposed by padding, which in larger applications might be significant.

(b) the verb `dtb` demonstrates a typical context in which a hook arises, when data has to be processed according to some function of itself, in this case "index of first blank" (`i.&' '`). Since the result of this transformation becomes the *left* argument of "take", the "reflex" adverb (~) is required.

(c) The verb "catalogue" (`{`) converts a simple list into a boxed list while retaining any fill characters, for example

```
   {usa
┌─────────┬──────────┬──────┐
│Florida  │California│Alaska│
└─────────┴──────────┴──────┘
```

so if application data is in the form of a simple list, but boxed list processing has been chosen, it can be applied "with catalogue". For example suppose that `MASC` is applied to `usa` following right justification using the same technique as in `dtb`:

```
   rj=.(|.~ i.&' ')"1                 NB.  Right justify character matrix
   MASC&{ rj usa
┌─────────┬──────────┬──────┐
│  Florido│Californio│ Alasko│
└─────────┴──────────┴──────┘
```

Returning to "amend" ...

The alternative to transformation with subsequent assignment is *amendment*. In the previous illustration of amendment, the new data and selectors were explicit nouns (9 and 1). Now a selector *verb* is required since the set of indices of *atoms* (i.e. characters 'a') which have to be changed is a function of the data itself.

An implicit assumption so far in the states problem is that the state names have been left justified. This assumption makes it simple to find the index of the final 'a' in each name:

```
   llc=:<:@i.&' '                     NB.  locate last non-blank character
   llc 'California   '
9
```

To transform these into indices to define atoms of the list usa, attach the index of the corresponding item, and then box each of the pairs of indices so obtained:

```
   ilc=:<"1@(i.@# ,. ilc"1)   NB.  index last character, e.g. <0 6 for Florida

   'o' (ilc usa)} usa         NB.  new data is 'o', selector indices are
Florido                       NB.  the result of  ilc usa
Californio
Alasko
```

Repeating the data-name usa in the above phrase is inherently unpleasing. However the specification of "amend" allows this to be tidied up by using "tie" to produce a gerund, which at the same time allows the replacement character to be parameterised:

```
   masca=:[`(ilc@])`] }       NB.  The 3 verbs in the gerund describe
   'o' masca usa              NB.  transformations which produce(left to rt)
Florido                       NB.     (1) the replacement data;
Californio                    NB.     (2) the selectors; and
Alasko                        NB.     (3) if required, transfn. of the old data
```

Without the gerund option, it would be difficult to accommodate "amend" in explicit definitions. A non-trivial transformation of the old data (i.e. verb (3) in the gerund) might be

```
   lcc=:3 : '(t-32*96<t=.a.i.y.){a.'   NB. Convert any l/case chars to upper
   mascc=:[`(ilc@])`(lcc@]) }          NB. Masculinise and convert

   'O' mascc usa
FLORIDO
CALIFORNIO
ALASKO
```

Using a boxed list raises a difficulty because it is necessary to do an "open" between successive indexing activities. To get round this an "amend" based verb can be defined to work at the item level and then applied to each of the items in the object:

```
   MASCA=.[`(<:@#@])`] }      NB.  selector verb gives index of last character
   'o' MASCA each USA         NB.  Change each last character to an 'o'
```

Florido	Californio	Alasko

Item Amend

So far, the adverbially qualified verb "selector}" has been used dyadically. It can also be monadic in which case } is called "item amend". The result has the structure of a single item of the right argument y., and its value is determined by selecting one of y.'s items to provide each item of the result. Each item in y. must therefore contain the same number of items. This sounds quite a mouthful, but a simple example which makes everything plain is the random answering of a multiple choice test. The data is five items each comprising 20 repetitions of the same character; the result of each execution is a further item, each of whose items comes from just one of the original five.

```
   ]mch=.|:20 5$'ABCDE'   NB.  Construct the character matrix which follows:
AAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEE
  (720$5) } mch          NB. (720$5)} is a monadic verb
ADDEBCEAACDABACDCEEC     NB. Hopefully you got something different!
```

When selection depends on the data, for example if it is required to generate the multiple choice responses in strict sequence, use a verb as selector. A verb to obtain repeating integers is

```
  rint=:({. | i.@(:)@$    NB.  Repeat row indices to length of #columns
  rint mch
0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
  rint} mch
ABCDEABCDEABCDEABCDE
```

What Use Are User-Defined Operators?

by Ian Clark

Dyalog APL permits user-defined operators. Apart from the header syntax, and the fact you need to type: `)ops` or: `⎕NL 4` instead of: `)fns` or: `⎕NL 3` to see their names, they are written and maintained just like functions. A lot of APLers I know claim to have no real use for them. But I make a great deal of use of user-defined operators and I wouldn't be without them.

I write a lot of code which is called by other APL programmers through a public interface, *i.e.* a pre-arranged collection of function (or operator) calls. Often I don't get to talk to the programmers calling my code. Eventually it gets maintained by someone else. If he/she ever gets to talk to me, then that's a failure on my part. APLers who have never written for such an environment don't appreciate what's involved. Unlike C++, the essential apparatus for large-scale code support doesn't come free with APL. But it's not difficult to organise.

Trawling through my old code, I notice my operators fall into the following (non-exclusive) categories:

1. variants of primitive operators
2. 3 or 4 argument functions
3. common code
4. function jigs

Now I don't have much use for 1, but I include it as a simple way of introducing the idea, plus exercising some terminology. First some implicit definitions...

An APL *function* `foo` takes a *right argument* `⍵` and maybe also a *left argument* `⍺`. You declare `⍺` to be optional by enclosing it in braces in the function header. Otherwise, when called, both `⍺` and `⍵` must be supplied. So these are the options:

```
[0]   z←foo ⍵
[0]   z←⍺ foo ⍵
[0]   z←{⍺} foo ⍵
```

An APL *operator* `oppo` takes a *left operand*, `f` and maybe also a *right operand*, `g`. Both `f` and `g` are thought of as being APL functions, but when `oppo` is called you can supply variable names instead, or even constants, whereupon `f` behaves like

a (function) argument, assuming the correct name class for its contents. There is no construct corresponding to {α} to show one of the operators is optional. So when called, both *f* and *g* must be supplied if both are specified. So these are the options:

```
[0]   z←(f oppo) ⍵
[0]   z←(f oppo g) ⍵
[0]   z←⍺ (f oppo) ⍵
[0]   z←⍺ (f oppo g) ⍵
[0]   z←{⍺} (f oppo) ⍵
[0]   z←{⍺} (f oppo g) ⍵
```

The result of executing an *operator* with given *operands* is actually an APL function, called the *derived function*. You can use it in a *functional assignment* like this to create a new function *foo*:

```
foo←(f oppo)
```

but the typical use is with both operands and arguments to return a value, just like a function call.

The result is that *oppo* behaves like a 3 (or 4) argument function. Except that one (or two) of the 'arguments' are really operands, and you can supply the name of a function or a functional expression which will appear inside *oppo* like a localised function...

```
z←x(f oppo)y
z←x(1∘+∘f oppo)y
```

Variants of Primitive Operators

Let's take as an example the (primitive) operator: (¨) {each}. Its behaviour is not to everyone's liking, because using it to apply a function *foo* to each element of an arbitrary list *y* is not what you'd expect if you're thinking of it as a *:For*-loop. If the list *y* becomes empty you might hope that *foo* would not get called, whereas it does — once, with an argument of scalar 0. Now, caring nothing for APL theology (which tells you infallibly that this is what you ought to expect and you should jolly well write *foo* to accommodate it) you decide to write a replacement: *PEACH* (polite-EACH) which does what you want, *i.e.* it doesn't call *foo* if the right argument *y* is empty.

```
      ∇PEACH[□]∇
[0]   z←(foo PEACH)y
[1]   z←⍬
[2]   :For j :In y
[3]   z,←⊂foo j
[4]   :EndFor
```

As a special case of `foo` we'll define a fn: `inc`, which simply bumps its (scalarised) arg by 1...

```
      ∇inc[□]∇
[0]   z←inc y
[1]   z←1+⊃y
```

Now, by putting a stop on line 1...

```
      1 □stop'inc'
```

You can verify that the next expression calls `inc` 3 times, the second one calls `inc` once, but the third one doesn't call `inc` at all...

```
      (inc PEACH) 2 4 5
      inc⍣⍬
      (inc PEACH)⍬
```

To demonstrate the existence of the intermediate *derived function*, assign it to a name like `foo`:

```
      foo←(inc PEACH)
      foo 2 4 5
3 5 6
      (inc PEACH)2 4 5
3 5 6
      (1∘+∘inc PEACH)2 4 5
4 6 7
```

You can put a stop on `inc` and see again the above calling behaviour. Note that alterations to `inc` are correctly reflected in `foo`, so it's not like ordinary functional assignment (which takes a snapshot of any function employed, so it does not keep in step with updates).

3 or 4 Argument Functions

One limitation of infix notation is that it allows functions to take no more than two arguments. Now of course you can call a function with any number of logically distinct arguments, concatenating them into a single nested right argument. The function can then take this apart by a threaded assignment.

Thus:

```
      x←foo u v w

[0]   z←foo y;u;v;w
[1]   u v w←y
```

However, when you're writing public functions to be called by others, who may not be people of your intelligence :-) it's nice to leave it to the interpreter to tell them they've not supplied an argument...

```
[0]   z←u(v oppo)w
...
      x1←u(v oppo)w
      x2←(v oppo)w
SYNTAX ERROR
```

Common Code

Throwaway code has no use for this. You can get away with replicating a function `foo` to make a slightly different one `goo`. But when you're writing long lasting code, which may be maintained by chimpanzees at some time in the future, sometime someone's going to edit `foo` and not realise that `goo` needs a similar change.

Here's a facile example of common code. Two functions, `next` and `prev`, are only slightly different. Here they just alter a global `XX`, but a more complex algorithm can be readily imagined with today's window applications. However instead of replicating `next` to create `prev`, both are defined on a common operator: `step`. Now of course this could be done without writing a user-defined operator (`step` could simply be a dyadic function) but in some cases (where `next` and `prev` are themselves dyadic) defining common code as an operator is a preferable solution.

In my experience, it's worth writing even the most complicated common-code operator to accommodate all the differences between `next` and `prev`, because it

does at least document these differences all in one place. In other words, never replicate common code, if you want it to stay common.

```
      ∇next[⎕]∇
[0]   next n
[1]   (+step)n

      ∇prev[⎕]∇
[0]   prev n
[1]   (-step)n

      ∇step[⎕]∇
[0]   (foo step)n
[1]   'XX was:'XX
[2]   XX←XX foo n
[3]   'XX now:'XX

      XX←1
      next 2
XX was:  1
XX now:  3
      prev 1
XX was:  3
XX now:  2
```

Function Jig

This is my favourite. The difference between this and the foregoing examples is in the programmer's intention. Here the operator is a bolt-on 'jig' for executing the operand. The algorithm does not logically need it. The function which serves as its operand is the thing that does the real work. The operator is there to modify its behaviour, like changing its environment, e.g. `⎕IO` or `⎕ML`. Its syntax intentionally makes it look like a bolt-on. It accepts the arguments intended for the original function and simply passes them through (Dyalog APL manages to achieve this without replicating the values in the workspace). In this example it allows recovery (sometimes) in the event of a WS FULL. In practice I find that some memory-intensive expressions may give WS FULL, but after garbage collection triggered by `⎕WA` it executes successfully. Such as this innocuous looking expression:

```
B←×C
```

(where C is usually a very large Boolean array, but we want to be quite sure B is Boolean).

```
      ∇wa[□]∇
[0]   z←(foo wa)z;rainyday;x;□TRAP
[1]   ⍝Apply z-filter: foo, with WS FULL handling
[2]   ⍝NB: foo must be monadic in this variant of: wa
[3]    □TRAP←1 'C' '□EX''rainyday''⋄□TRAP←0⍴□TRAP⋄→AGAIN'
[4]   ⍝trap (1=) WS FULL
[5]    rainyday←1000⍴2 ⍝--safety buffer to release on WS FULL
[6]    z←foo z
[7]    →0
[8]   AGAIN: ⍝--WS FULL: try again...
[9]    hint'...garbage collecting, please wait...'
[10]   x←□WA ⍝--collect garbage
[11]   □TRAP←1 'C' '□TRAP←0⍴□TRAP⋄→EX' ⍝--trap WS FULL, timeout etc
[12]   z←foo z ⍝--try again
[13]   →0
[14]  EX: ⍝--admit failure...
[15]   ⋄ ⍝--crash, in this instance.
```

Note the use of `rainyday`, a so-called 'rainy-day fund' of emergency memory. Some WS FULL conditions may lock up nearly all ws memory, preventing a recovery function from doing even the simplest housekeeping. A rainy-day fund can let it soldier on. To use this operator, replace the WS FULL-prone function call, (say):

```
z←foo x
```

with:

```
z←(foo wa)x
```

British APL Association Vendor Forum
May 21st 1999, at the RSS, following the AGM

notes by Stefano Lanzavecchia

While waiting for the meeting to start, the attendees had a chance to look at the latest experiments of Causeway's Adrian Smith, this time regarding the first attempts at integrating HTML with vector graphics, using Microsoft's *Vector Markup Language* (VML), as a platform to output the graphics of *Rain*. The technology is still in its infancy; so far only Internet Explorer 5 and Office 2000 implement it, and the W3C group has not agreed on a standard. Yet I am sure we will hear again about this; readers with IE5 can see the examples and read the source code on *www.causeway.co.uk/vmlsamp.htm*.

Eric Baelen of APL2000

Eric Baelen, former CEO of APL2000, started his speech with a joke, to warm up (or maybe cool down the overheated room) the audience. There are three main differences between British people and Americans: in Great Britain, *English* is spoken; when staging a world championship, Great Britain invites people from *other nations*; when meeting a head of state, the Brits only have to go down on *one* knee. Luckily enough the projector and the laptop were quickly prepared so that the presentation could start with a series of slides off our favourite PowerPoint.

Eric described the latest acquisition of APL2000/LEX2000 by Cognos — "dramatic" news, but not in negative sense. Cognos is quite a large company, valued around 300 million dollars with some 1600 employees in 40 different countries, its expertise being in what they call "business intelligence". It's quite easy for an APLer to understand what they mean, by looking at their statement of intentions: to turn data, large amounts of data, into information. A task perfectly suited to APL, and Cognos decided to take over the old Manugistics to exploit the great capabilities of APL in the field. Cognos is now developing, with the aid of the APL2000 team, a new product, completely written in APL.

Knowing the interests of his audience, Eric moved quickly over the company in general, to focus his attention on the upcoming release 3.5 of the APL+Win interpreter. It was good to see that it was a live demonstration based on an internal experimental version of the code, which should be frozen in mid-June. The list of enhancements is quite long, yet none of them is so critical to have suggested that it was time for a change in the major version number. Most of the

features added or improved are attempts to make the life of a programmer easier. In this summary, I will follow Eric:

- session log: the session can be saved in a file with many available options to control how these files are created;
- the long file names of Win95/98 and NTFS are now supported;
- value tips: little windows with the content of a variable pop-up when the mouse points in the session (and in the editor?) to the name of a previously assigned variable;
- both single quotes and double quotes can be used in a string to identify the quote, removing the need to double them up; does this mean that it's now impossible to enter literal strings containing a double quote? *[See below]*
-)SAVEOVER removes the need for the normal sequence)WSID wsname, followed by)SAVE;
- the help key (F1) pressed on a token in the session (and the editor?) opens the online help file at the relevant page;
- stops can be set on line [0] of a function to inspect the explicit result just before the execution of the function is abandoned;
- 2 ⎕*AT* is implemented to inspect the Fix time of a function
- the left argument of ⎕*DEF* can be used to force the result of ⎕*AT* to a particular date and time.

[The issue of single and double quotes was raised on comp.lang.apl; the following response should set readers minds at rest here ...]

```
------------------------------------------------------------
From: Jim Weigang <jimw@chilton.com>
Newsgroups: comp.lang.apl
Not to worry--

      'The enhancement is "upwardly ','compatible".'
The enhancement is "upwardly compatible".

      "The enhancement is 'upwardly ","compatible'."
The enhancement is 'upwardly compatible'.

The outer quotes establish the delimiter, and within the string the
"other" quotes are treated as ordinary characters.  Nice enhancement!

                                  Jim
---------- ------------------------------ -----------------
```

More items were on the list but Eric decided to skip over them, due to the limited time, and he concentrated on the enhancements of the GUI controls available in APL+Win. Following the past tradition, most of the new features introduced by

Microsoft for their common controls (i.e. ListViews, TreeViews, ProgressBars) can now be accessed from APL:

- **TreeViews** can have checkboxes on every branch, leaf;
- **ListViews** can have checkboxes on every item, can display gridlines and support the highlight-on-over single-click-select typical of the new shell of Windows 98 and Windows 2000 in WebView mode;
- **ProgressBars** can be coloured, can be smooth instead of composed of blocks and can be oriented vertically;
- there is a brand new **Calendar Control** to allow input of dates; particularly interesting is the fact that it can be customised quite heavily for special needs;
- **Rebars** can be used instead of the now old-fashioned ToolBars, and, surprise, they can be made to float and dock under user or program control: time for a reaction from Dyadic...
- users of the **Printer** object will be delighted to hear that it now supports Orientation among the properties that are exposed. More will be added in the future;
- the **System** object generates events on changes in the environment.

A major extension in the language, which Eric did not have time to discuss but only to mention, is the **OnNew** event on the creation of user-designed classes. If I understand correctly it permits a developer to code new objects directly in APL composing and extending the behaviours of existing objects. It sounds quite exciting.

⎕WCALL now supports external functions requiring structures to be passed as arguments, allowing them to be represented as standard APL nested arrays. One doubt remains: can they be nested at any depth to simulate structures pointing to other structures?

The accuracy of *⎕MF* has been enhanced from the old 1/18th of a second to the one offered by the high performance timer provided by the modern CPUs.

The utilities shipped with the interpreter now include a Browser for resources in DLLs or executables (quite handy to extract bitmaps and icons for use in other applications) and a more advanced Workspace Explorer.

This concluded the presentation of the new upcoming release 3.5. We really look forward to laying our hands on a review copy as soon as it is available.

Eric Baelen also wanted to share with the audience the plans of Cognos/APL2000 to tighten the link between APL and the World Wide Web. He showed a web site called *OpenHere.Com*, which is a portal site, like Yahoo to mention one, whose main characteristic was to have all its server side software written in APL. Despite its recent genesis, according to Eric, it's already one of the 10 largest search sites on the Net. Born as a simple technological experiment, it turns out to be a good ground to test and possibly patent ideas.

To close his show, Eric had a surprise for all of us: a videogame written in APL. Using calls to Microsoft DirectX technology, it features a map of the world on which a colony of rabbits lived happily in the States, when the fierce race of the ducks started expanding. Purpose of the game: to protect the rabbits by guiding a super-hero around in his hunts for the evil ducks. I quote Eric: "The ducks usually beat me, so I asked to have programmed a key combination which kills all the ducks in one go. It's good to be the boss". *Applause.*

Pete Donnelly of Dyadic Systems

Pete Donnelly from Dyadic took over. His opening quote is a curse for one of his favourite developers to whom his laptop had been lent with the result that some things had stopped working properly. His presentation did not contain the introduction of any really new features, but did show a couple of interesting enhancements. To make his point he started by recalling the story of the interpreter: some of the concepts pioneered by Dyadic were received at the beginning quite coldly; namespaces, for instance, did not generate a lot of positive feedback to start with, but now all those who tried them would not be able to live without. Especially since the cloning mechanism of namespaces has been optimised. The same happened with dynamic functions, whose fans are increasing daily. So, Pete said, it's no surprise that the recent introduction of multithreading has not yet changed the way of working. He believes, and I am of the same opinion, that it's only a matter of time before the APL community discovers the true power of a multithreaded APL.

Pete proceeded with a short demo of what in practice is a multithreaded APL. The $\square SI$ stack is a tree and can be split at any point with the application of the new operator "&" (ampersand) that any *nix user will recognise. The new branch is executed in parallel with the others. The interpreter can be instructed to spawn a new thread for callbacks associated to GUI events. The demo of the bouncing duck can be written in a very elegant way using a separate thread for every duck, and this makes John Scholes very happy.

A recent enhancement to the multithreading capabilities of Dyalog APL is the ability to execute asynchronously a ⎕*NA* call. Because of the scheduling mechanism implemented in the current version of Dyalog APL, a long external call (like a bulk update of a database using SQAPL), could prevent the other APL threads from continuing their processing. Now it's possible to declare in the ⎕*NA* specification that the function should be executed in a separate native thread (as provided by Win32). Some extra care should be applied, but the results can be quite amazing. On a multiprocessor machine, the second CPU can effectively execute a bulk update while the first one continues the execution of other parts of the APL program.

Another recent extension is the possibility of creating in-process **OLE servers.** The main advantage is that the protocol used by the client to communicate with the server is much faster. The disadvantage is that the DLL (such is the aspect of an in-process server) is loaded in the process space of the client, which means that it can bring down completely the client, in case of a serious crash, and that if multiple clients connect to the same server, there is duplication of data in memory. Also, at the moment it is impossible to do any debug of the server. In many cases, though, the pros balance and overweigh the cons.

Another enhancement showed by Pete is for the **Grid Object,** which has been extended to allow the locking of arbitrary rows and columns for scrolling. The mechanism is very generic and will surely find many uses. A further extension allows users to specify gridlines in detail — line colour and weight may be set for any number of gridline styles which are then applied to the individual horizontal and vertical lines.

The next topic of the demonstration was for ActiveX controls. This repeated the demonstration first shown at APL98 in Rome.

Finally, to show a first example of a real application that benefits from multithreading, Pete demonstrated the internals of the APL-based web server, which runs on an experimental site at Dyadic.com. The aim of the server is to provide a platform on which to develop and run APL web-based solutions. Pete recommended the persons interested in the deployment of web-based applications in APL to contact Dyadic Software. I also recommend all the owners of a copy of Dyalog APL to study the source code for the server that is distributed with Dyalog APL 8.2 and can be found in APLSERVE\server.dws.

David Leibtag of IBM's APL2 Group

After a short break for the coffee, the podium was left to David Liebtag, responsible for the development of APL2 in IBM. All the way from California, he thanked the organisers for the opportunity that was given to him to discuss the present and the future of APL2. I really feel like saying: thank you, Dave, for giving us some of your time. Admittedly it had been quite a while since the last time he had come to Europe. Two of the main topics for his presentation: Service Level 4 of the Windows version of the interpreter, and a highlight on what's-next in IBM for APL2.

Until before the release of Service Level 4, IBM had tried to catch up: Dave explained that Service Level 4 is the first release that, based on the feedback from the users, is devoted to making the development environment more comfortable and productive. Here's a small list:

- show the fix time for the function in the status bar of the function editor;
- position the cursor in the line indicated by the error message;
- break points: in a way similar to what can be done in Dyalog APL, for instance, break points can be set to interrupt the execution of a function.

Various enhancements to the interface components:

- the Rexx processor allows APL to call directly the built-in functions;
- support for SQL/DB2 has been extended to include BIGINTs, LOBs, and to tolerate a WITH clause preceeding a SELECT;
- there is a new efficient way of communicating with the GUI, no longer based on the expensive shared variables, but on a GetProperty/SetProperty pair of functions;
- the GRAPHICS package now features additional marker types, new line/dash styles for the Win32 platforms, optional box corner radius.

The release is available on the IBM web site.

So far so good. But what Dave really wanted to discuss was the future of APL2. The philosophy that drives the development is to enhance the interfaces available to and from APL2, to allow developers to easily leverage existing code and integrate more thoroughly with the rest of the environment.

First of all: integration with Windows. David is working on a keyboard handler that will allow input of APL characters, using the key combinations allowed by

all the popular interpreters. We were given a live demo: really quite impressive, if it wasn't for the lack of support for WinWord, which uses its own low-level keyboard handler. David says that when the little application is finished, it will probably be freely distributed to the APL community.

Next on the agenda, was the disclosure of a project Top Secret, regarding the High Performance Link (HPF) in use on the mainframes. A change is being discussed, to reduce the overhead of function calls, which is not backward compatible. If you are interested in knowing more about this, you can contact David Liebtag at *liebtag@us.ibm.com*.

Another planned extension for the interpreter is the ability of sending and receiving APL arrays in native format. Dave asked the other APL vendors to discuss the format. My opinion is that since Dyalog APL already supports this feature, maybe that should be the starting point.

The end of David's presentation also coincided with the end of the meeting. A group of the attendees and the three kind speakers rejoined in a neighbouring pub to enjoy the end of a good and successful afternoon.

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.

Teaching Functional Programming with J

from Neville Holmes

Coming to Tasmania to teach in a new degree course, I looked for opportunities to insinuate the APL mode of thought into some of the course. It was not easy. Eventually, struck by the idea that J's tacit programming was, or could be seen as, a delightfully pure form of functional programming, I put together a unit of teaching called "Computation and Functional Programming" for Honours students. There was little interest for a few years, but eventually word got around the students and it is now very popular. The students are required to undertake a project of their own choice, but they are constrained to using J in tacit/functional style. *Vector* is publishing some of these papers, the first of which you can find in Vol.15 No.4 (Andrew Towers, page 50) and the second follows on page 95 in this issue.

An APL Unicode Font

by Phil Chastney

The Vector website has another APL font for downloading. The font is called SImPL.TTF and, as the file extension suggests, it is a TrueType font for use on the PC. (*See* www.vector.org.uk/resource/simplttf.zip)

Some Details

Describing the thing as an "APL font" is a little bit of an over-simplification. One of the design aims of the font was to provide a complete set of all known APL symbols, plus sufficient characters to allow prompts, comments, *etc.*, to be expressed in every European language known to be in current use. Basically, that means the Latin, Greek and Cyrillic alphabets, plus accented and variant letter forms as required for other European languages using these alphabets.

This is not going to fit into the usual 256-character strait-jacket: the lowercase Latin letter "a", for instance, exists in 16 variant forms — so far. By limiting the spec to modern European usage, we have excluded variants used in phonetics, African written forms, Vietnamese, ...

If an 8-bit character code is too limiting, the obvious thing to do is go for 16-bit Unicode, and the SImPL font is a properly constituted Unicode font: each character has been given its proper Unicode value, and may be viewed in Unicode order using the Windows "Character Map" applet.

It is obvious by this stage that I am addressing a small audience, comprised of those internationally-minded APLers working under recent versions of Windows.

In that case, then, why bother? And in particular, why bother designing a new font? The reasons are partly idealistic, partly practical and partly historical.

The Reasons Why

Hitherto, hard-copy APL listings have always been a bit of a nuisance. On an individual basis, solutions could be found using a locally-attached printer, but life became a little more difficult when using networked or mainframe printers. Success here depended almost entirely on the goodwill of the people controlling the network or mainframe. Were they, for instance, prepared to set up a comb printer or a band printer capable of switching to APL? Where this goodwill was

lacking, life become difficult. On one occasion, a user having trouble with a departmental printer was told the problem was caused by his attempts to print APL. In fact, the connector at the back of the printer was falling to pieces, which was soon fixed, and is the reason I always carry a screwdriver with me nowadays. (This did not fix the "attitude" problem, however.)

Now that the world has switched to Windows, Adrian's APL2741 font has been a real god-send, but different character sets are required in different situations, and translate tables are often required between the originating processor and the target printer, because it is difficult, and possibly ill-advised, to re-arrange the characters within the font.

It is also difficult to extend the character set to include new items: I recently found myself unable to print the broken vertical bar when I wanted it. Back in the DOS-only days, a personal desire to include the Greek lambda in the APL character set meant hacking the character tables included in the APLFONT utility provided with APL*Plus, for proper screen display. This was not too difficult, but editing the DeskJet font for proper printing was rather more tedious. Both worked fine (eventually), but extensions to the character set no longer seemed so desperately important. Nevertheless, I managed to hook up a system which could display on-screen any item from a set of approximately seven hundred 9-by-16 characters.

The rest of the world, however, had moved on, and it was time to catch up. But how? I was working in an international environment, using DOS, Windows and Unix. The "national use" characters defined in Unix looked as though they might be useful but, to take an actual example, if a Brit in Luxembourg receives a file from a German in Madrid, what can we assume about national character use? In truth, the national characters are no use whatever in the international arena.

The DOS code was recycled. It was not possible, at that time, to have 1024 characters in a single TrueType font, so the thing was re-organised as four separate "pages" of 256 characters each, one page for Latin-1, another for additional accented Latin letters, a third for Greek and Cyrillic, and finally a pageful of APL and other symbols. A stream of Unicode values was mapped onto the appropriate codepoint of the appropriate font (where possible). Messy, but it worked. I was particularly pleased that I could display all three scripts simultaneously, in the same font.

This was my first experience of building a font from scratch. The result displayed OK, even at low resolution, and printed nicely on my DeskJet, and that was enough for the time being. And now I could add in all the additional characters I wanted.

But the world has moved on again. Recent versions of Windows have Unicode fonts. Check out the Character Map applet, and you will see that standard stuff like Arial, Courier and Times Roman now includes Greek and Cyrillic scripts as standard.

But no APL.

Even Bitstream's Cyberbit font – a magnificent achievement which includes everything you are likely to need for modern Chinese, Japanese and Korean usage – does not trouble itself with APL. (In fairness, it should be noted that Cyberbit does not trouble itself with mathematical symbols, either, so we should not feel unjustly discriminated against.)

So – back to the original question: why design a new font? Answer:

- because I want to be able to print APL characters (*all* APL characters);
- because I want to be able to add in other characters;
- because I want to be able to print out prompts and comments in other languages;
- because I want a to do this with the same font, for consistency of appearance;
- and, finally, because there is no font yet which meets those criteria.

The File Format

It might seem that the choice of PC TrueType format was a foregone conclusion. Not so. Well, not entirely so. Certainly a PC .TTF is the most convenient way of displaying and printing under Windows, but it is not the only way. Adobe Type1 fonts are preferred by professionals. Robin Williams, in her book "Bossing Your Fonts Around", is quite scathing of TrueType fonts, and recommends sticking to Type 1 fonts, where possible.

The problem is that TrueType fonts are not well-behaved, when used with high resolution type-setting equipment. Three possible reasons spring to mind:

- TrueType uses quadratic splines, whereas Type1 uses cubic splines. Quadratic splines are easier to calculate, which means faster rasterisation, but the fonts themselves may be more "brittle". If you have ever had a font which would print correctly only up to 24 points, say, or only above 24 points, or would rasterise correctly only for some characters, then you will know what I mean.
- The TrueType standard is incomplete. This may mean that interaction between settings is not well-defined.

- The TrueType standard is obscure or ill-defined. The manual for Fontographer 4.1 states, ".. there are a fair number of fields that we don't really know the purpose of .." so there is little chance of an amateur succeeding here.

I once had a TrueType font which would crash the system completely – requiring a power off/power on reboot – if I so much as clicked on its name. Don't ask me how or why it did that: I was simply relieved when I finally managed to delete it. Also, I know of one case where a photo-typesetting agency lost a contract when a client found out they had allowed TrueType fonts on the premises.

Notwithstanding all that, TrueType is a convenient format for distribution, easy to install, and (provided Fontographer is trusted to provide sensible default settings) can be made to work more than adequately at sizes from 8 to 144 points.

The Character Set

Languages are disappearing. We are losing diversity, and this is immensely regrettable. It would be nice to be able to prepare text, display windows, etc., in all languages, but the resources required prohibit this, so we must settle for the lesser goal of supporting all modern European languages. I do not know a word of Welsh, and it is unlikely that I shall ever bother learning Welsh, but I would not wish to see it disappear. This new font, therefore, contains all the accented characters required for Welsh. And Catalan. And Macedonian. And . . . (And, as it happens, all the characters necessary for Pin-Yin Chinese and Romanised Sanskrit.)

At the time of writing, the font contains about 1000 characters. For Unicode devotees, this includes the whole of Latin-1, the whole of Latin Extended A, much of Latin Extended B, very little from Latin Extended Additional, and whatever seemed necessary from the Greek and Cyrillic blocks.

There are no less than 5 versions of the A-to-Z of the basic Latin alphabet:

- uppercase A-Z
- small caps A-Z:
  Small caps can be faked by simply switching to a smaller font size, but this is not entirely satisfactory because the stroke weight and any serifs are thereby scaled down, when what you really want to do is maintain stroke weight, maintain serif size, but reduce the overall height. In essence, you want to scale down the vertical distances between horizontal black bits; in practice, you have to redraw each character. The small caps are not immediately available to users, although Windows provides a GSUB (for "glyph substitution") mechanism which will allow them to temporarily replace the lowercase letters.

- lowercase a-z:
  The usual 26 character complement is extended by the inclusion of the Eth and Thorn characters from Icelandic, the Yogh from Middle English (which is also used for the Ezh character found in Lappish) plus some alternative forms: a loop-tailed "g", a long "s" and a round-tailed "y". (I was delighted to discover that Eric Gill also used to do something similar in his fonts.)

- tiny capitals:
  The tiny capitals are not intended for general use, but are used to construct the "control pictures" (Unicode values U-2400 *et seq.*) which allow control characters to be represented literally. The control pictures are omitted from most Unicode fonts, but can still be useful to those of us struggling to communicate with external modems.

- underlined capitals A-Z:
  The atomic vector for IBM mainframe implementations contains an underlined uppercase alphabet, in addition to the normal uppercase and lowercase alphabets. This is more than a little odd, when more useful characters could have been included in the codeset, but if that is what IBM decides, then these characters must perforce be included in the SImPL font.

Modern Greek typography seems to be in a state of flux. The most common diacritical marking in modern Greek is the "tonos", which denotes stress, and this is variously represented as an acute accent, a caron, or a downward pointed triangle, among others. I have used the acute accent with lowercase letters, and an apostrophe with uppercase letters, because this seemed to be the accepted practice when I started drawing this font. There may be a problem here: classical Greek used apostrophes as breathing marks, to indicate whether an aspirate (an "h" sound) occurred before any words beginning with a vowel. Recently, however, I have seen breathing marks used in modern Greek. If this practice becomes widespread, it may be necessary to redraw the uppercase vowels.

The main problem with Cyrillic is likely to be missing characters. The script is used for such a variety of languages beside Russian, that it is more than probable that something has been omitted from one of the minority languages.

There are some alternative characters in the font. The ampersand exists in two forms. The ess-zet ligature exists in three forms, the one normally available being a fairly literal ligature of a long-s with the normal lowercase s, rather than a beta character which has been stood in the sun for too long.

The Numerals and Currency Symbols

The obvious set of digits 0-9 is provided, in two forms. Digits and currency symbols are normally drawn to the full f-height. (The letter 'f' is usually the tallest

lowercase character, so the term "f-height" denotes the highest point reached by a lowercase letter. Similarly, the "g-height" denotes the lowest point, and the "fg-height" measures from top to bottom.)

The habit of drawing digits at the full f-height is a comparatively recent introduction. The logarithm tables I had at school (approx. 3 million years ago) used the so-called "old style numerals" or "lining numerals" or "x-height numerals". (Typographers seem largely unaware of the difference between a "digit" and a "numeral".) Some people prefer these old-style numerals, finding them easier to read, so SImPL provides both types - f-height and x-height.

Fractions have been a real pain. This is not the time to go into all the detail, but the fixed-width chosen for this font does not leave much room for two digits and a slash. All the fractions included in Unicode are provided, without slashes, because that is what the Financial Times does in its pages of share prices. Traditionalists are catered for by an alternative set of fractions using a slash. There may be a call for yet another set, with a shorter, shallower, slash, and any such call will be acknowledged, noted and probably ignored until I am happy with the present offerings. There are no facilities for constructing your own fractions.

The Symbol Set

The obvious items are included for punctuation and accents. I have also included whatever arrows, mathematical operators and miscellaneous technical symbols seemed appropriate. (The APL characters are found at the end of the section of "Miscellaneous Technical Symbols".)

My choice of symbols may strike you as idiosyncratic, not to say egregious. Certainly, the Unicode set of APL symbols strikes me as odd. While I have drawn glyphs for each of these characters, I cannot help wondering who uses them, what for, and are they really necessary. Nor is it clear why we need to duplicate nand, nor and circle-star. I might add that some of these glyphs are visually unappealing, though they do not have to be quite so ugly as the samples shown in the Unicode standard.

Be that as it may, if the font is missing any other characters from other APL dialects, they can be included with little difficulty (though access without a Unicode value may be difficult).

For backwards compatibility, the font includes the box drawing characters included in previous APL codesets, although it seems unlikely that anybody in a position to utilise this font is still drawing boxes using these characters.

Font Design

Traditionally, APL has used a Courier Italic font. Actually, Iverson's original text did not: the association between APL and Courier Italic started with the 2741, IBM's golf-ball printer.

My interest in APL was sparked in the first place by the font it used. To someone struggling to express his ideas in a 48-character code which did not even have lowercase, it seemed so expansive, wild and free. Clearly I am not going to rush to break that association.

Let us review, then, the characteristics of Courier:

- First and foremost, it is a fixed-width, or monospaced, font. That is to say, each letter occupies precisely the same amount of horizontal space, so the letter "m" has to be squashed up, while the letter "i" has to be stretched out. While not necessary for printing, it helps screen display if we add the restriction that all characters require exactly the same amount of vertical space.

- It is a slab-serif design. This helps readability. The serifs on the lowercase "i" require more ink than the basic letter, but they help to lead the eye in and out to the next letter. This principle seems to be easily forgotten: most Courier versions of Greek leave the iota standing alone in a huge white space, which slows down the way the brain recognises a word.

- The x-height (the distance from the baseline to the top of a flat-topped lowercase character without an ascender) is a relatively high proportion of the f-height. This, too, aids readability.

- Unusually, the cap height is lower than the f-height, and the uppercase M is approximately square. There is no stress: vertical, horizontal and diagonal lines all have the same width. Although distinctive, these three characteristics do not bear on the choice of Courier as a model for the SImPL font.

It is possible to use proportional fonts for code listings. I have used PostScript routines which output C listings in Helvetica or Times, and been generally happy with the results. The vertical alignment of the opening and closing braces goes awry occasionally, double and single quotes are not easily distinguished, and it is difficult to tell how many spaces are contained in a string but, all in all, the results were quite acceptable.

There are still reasons for preferring fixed-width fonts, however, especially with the rather denser (typographically denser, that is) code encountered in APL. Vertical alignment is less critical, and it is important that the operator symbols do not visually overpower the symbols for the operands. Where vertical alignment is important, in multi-line comments, for instance, this is more easily achieved with a fixed-width font. Also - and this is of interest to the font-designer and the advanced user - it is easier to place diacritical markings (accents) in a fixed-width font.

The SImPL font, then, is based on Courier, though there are a few differences.

In the first place, it is a semi-bold font. The Windows Courier font is a Light font, *i.e.* the vertical strokes are relatively thin. I preferred the Courier font supplied with the printer, being thicker and blacker and easier to read, and this is the effect I aimed for with SImPL. (This may have been a mistake. Some of the Cyrillic characters, and some ligatures, proved difficult to fit into the prescribed width, and a lighter stroke weight might have made this easier. Redrawing everything, however, is not an option at this stage.)

In the second place, it is an upright font. The 2741 APL golf-ball used an Italic font for the letters, and left everything else upright. This was rather splendid - it gave the code a sense of urgency, somehow, and I have always liked the effect. A true Italic font, however, requires a completely redrawn set of characters. Compare Times Roman with its Italic version. Much that is labelled "Italic" is more properly described as "Oblique". Oblique, or "slanted", characters can be generated by applying a simple mathematical transform to the outline of the upright character, so SImPL provides outlines only for the upright characters, and leaves the user to generate oblique versions at runtime, should these be deemed necessary. It would be perfectly possible to write a listing program which slanted those letters used in the code as identifiers, and left upright those letters appearing in comments and literals.

Accents and punctuation are rounder and blacker. Courier proper uses a square "dot" over the lowercase "i", and then uses the same square dot in the semicolon. SImPL punctuation uses more rounded forms. Many fonts have quite insignificant diacritical markings, but not SImPL.

Most importantly, the uppercase A has a flying serif. In this respect, SImPL differs from APL2741.TTF and follows Courier. Intriguingly, the APLPLUS.TTF claims to be based on a monospaced version of the Rockwell font, yet Rockwell's "A" has two flying serifs, while APLPLUS follows Courier in having just the one.

I have gone so far with serifs as to add them to the "clicks" found in Latin Extended B. These characters are used in Khosa and Zulu, so if there are any Zulu APLers reading this, and they think the result looks weird, they will have to let me know, and I will change it.

The Font in Use

The following is an extract from Bitstream's documentation:

> To create documents that use any of the characters in Cyberbit, you need an application that supports Unicode, and you need to know how to enter the Unicode character codes in your application. Otherwise, you're limited to using the characters in the languages that Windows 95 or NT supports.

Quite! To paraphrase:

- Applications support for Unicode is desperately thin.
- That is your problem.
- If you cannot find what you need, you will have to write it yourself.
- At least, you now have a font to work with.

Reviews

The objectives were

i) to support all modern European languages,

ii) to support APL,

iii) and to do so in a visually acceptable manner.

How well have these objectives been met?

I was in the bath, after an evening spent adding some obscure accented characters to the font, and was just sliding beneath the water level, with a smug satisfied feeling of a job completed, when I felt like I had been hit on the head. There is a great gaping hole in the European coverage: no Yiddish. Yiddish uses the Hebrew script (in fact, they were using the Hebrew script before modern Hebrew was invented), but its vocabulary is 80% German, and although declining, it is still in current use and undeniably European. This is a problem. Most Hebrew fonts are script or sans-serif: a monospaced font, similar to Courier, appears to be an impossibility. Stretch some of those characters too far horizontally, and they turn

into different characters. The only solution appears to be a heavy, blocked, unstressed font, with white space showing around the narrow characters. If anybody who knows of a Courier-style Hebrew font, I'd love to see it.

Having thus been snapped out of my complacency, I checked what else might be missing. Europe, I was always told, ended at the Urals. But where is the southern boundary? If it is the Caucasian mountains, what languages qualify as European? Georgian and Armenian scripts could be included, I suppose, but the work involved seems intimidating.

There may be other smaller omissions, but it is difficult to be sure.

To quote Bitstream again:

> Bitstream has "delta-hinted" its most popular text sizes (10 and 12 point, at screen resolutions of 96 and 120 dpi; and 14 point, at a screen resolution of 96 dpi).

Well, you do not get that level of service here. The font looks OK on my screen at 10, 12, 24 and 36 points, except that "q" and "g" looked a bit untidy at 24 points. I have manually edited the hints, and that seems to have fixed it. You may get different results on your screen: this will depend on what resolution you are using, the size of the screen, the dot-pitch of the phosphor dots and (possibly) the software driver. I have absolutely no intention of editing bitmaps for improved screen display - the amount of work involved is horrific. In addition, the font prints legibly at 10, 12, 24, 36 and 72 points, without any obvious nasties. That, I am afraid, is the extent of my testing.

The letter forms in this font have been developed over a number of years, and as a result some inconsistencies have crept in. These are most visible in the Greek section, where artistic licence took over at one stage, and the strict design principles of the Courier font were set aside. This area needs revisiting. Likewise, some of the Cyrillic forms could be a little more relaxed. On the whole, though, I am not dissatisfied with the results.

The Future

Some of the letter shapes could be improved, and some of the letter forms could be better aligned with each other. This will happen slowly.

The selection of pre-formed composites (*i.e.* a base character plus diacritical marking(s) available under a single Unicode value) will be slowly extended.

The Unicode Standard says, with reference to the Latin Extended Additional block:

> The characters in this block constitute a number of precomposed combinations of Latin letters with one or more general diacritical marks. ... Each of the characters contained in this block may be alternatively represented with a base letter followed by one or more general diacritical mark characters found in the Combining Diacritical Marks block.

This might seem to suggest that a half-decent rendering machine would not need these pre-formed characters, but would be able to generate them on-the-fly, from the base character and its diacritics. In fact, placing accents is not easy to do under program control. Far better to resolve the incoming stream of base+composite into a single Unicode codepoint, and then access a fully-formed glyph from the font. If support for Vietnamese is to become a reality, for instance, it will have to be done this way.

Yiddish, Georgian and Armenian are best regarded as being on hold, indefinitely.

You are entitled to a contrary opinion on my aesthetic preferences, but it is unlikely I shall take any notice, though I am open to input on other matters.

Details of missing characters will always get a swift and sympathetic response. If, for instance, you prefer to write all your comments in Gibberish, but find yourself handicapped by the absence of the dotless-j-bar-with-hook, then one will be provided instanter.

The font could be made available in other formats. A Type 1 version is no problem; I believe a Macintosh version is a possibility, but I have never done one. There will not be a separate Ghostscript font.

The real need, now, is for utilities:

- a basic Unicode interpreter, capable at least of displaying the characters in this font;
- some form of access to those font elements (e.g., small caps) without a Unicode value;
- some way of toggling alternative character forms, such as old-style numerals;
- some means of glyph substitution, so that the font can be extended or partially replaced with glyphs from another file.

Any volunteers?

GUI and Closed Systems

by Anthony Camacho

Many systems are designed to enable office staff to do a restricted number of things to a set of master files. Such systems may have some of the characteristics listed below. In particular the system is *closed*: that is, the operators cannot do things which were not foreseen by the designer. For such systems the effort of producing a GUI front end is a waste of time.

Here is a list of features of a closed system that detract from any benefits that GUI might bring.

1. A Well Defined Menu Structure

The Windows style of GUI is incompatible with a well-structured menu, especially one which is designed to limit the places where records are changed so as to give the least danger that records are corrupted, lost or damaged during file interactions. For example when files are accessible to more than a single operator if, at the top level of the structure, the choice has to be made between viewing record details and amending them, then file locking procedures can be limited to amenders; viewers can be given unlimited access to records without any worries. File locking for safety when there are multiple paths between windows with options on them gets very complex.

2. Simple Help Provided Automatically

Simple menus where the choice is made by moving the highlight up and down a list of choices are very quick to operate and allow provision of a panel of help different for each choice which is displayed just while the choice is highlit. The rapidity with which help panels can be displayed even on quite slow Pentium PCs, is remarkable. Even a list of choices which has to be scrolled because there are more lines than the screen can accommodate, can have the panels displayed fast enough for any operator to feel there is no delay. There is an example of a simple function to do this in the Appendix. If the menu structure is well designed and the panels of help well thought out such a system can be operated after very little training. Such a design is inherently context-sensitive. It tells you what this option *is* rather than the Windows help which tries to find out what it is you are trying to do out of such a huge choice that finding what is relevant is often really difficult. In a closed system the range of things one can do is small. The aim is to avoid the delays inherent in the Windows help system and never to cause an

operator to have to press an extra key even when learning the system. Training costs otherwise can be quite significant.

3. A Restricted Set of Operations at any Point

The aim is to reduce the damage that could be caused by an erroneous key or click and to keep the design simple; less to program and less to learn. There should be a standard key for backing out of any situation without making changes (it is <escape> in the appendix).

4. File Operations that Protect against Partial Amendment

There are a few screens displaying details of records for amendment, so the places where amended records are written to files can be limited and controlled so as to reduce the danger of inconsistency between the state of one file and another to the absolute minimum. For example, when a system holds several indexes to give quicker access to some of the information, the chances of a glitch leaving some indexes saved and others not saved can be reduced by putting the instructions that make the saves in as short a sequence as possible. It is even possible to set a flag at the beginning of the sequence and unset it at the end so that if there is a power failure part way through, on restart the system can tell that this process was interrupted and the records of it should be checked.

5. An Instantaneous Response to any Command

(or a screen indication of action if the process triggered is a long one)

Programs that operate in character mode using character screens have a great speed advantage over those that operate through the Windows or any other GUI front end. Even with a really old and simple interpreter such as APL*PLUS/PC (I am using version 8.1) the speed of screen writing and reading is so quick one cannot fault the response times. On some legacy systems there were processes when the program displayed a message saying, in effect, "Be patient; the computer is sorting the alphabetical indexes". On quite a slow Pentium one has to pay very close attention if one is to see a flash of colour when the message is displayed; there is no chance at all that anyone could read it. Similarly, when cycling through 8000 components containing packages in a search for records matching a specified set of characteristics, it is a very easy matter to show progress on a bar chart, and the extra time for the whole process is not perceptibly increased by providing the progress bar. [Sample bar chart program in Appendix] So there is no problem ensuring that the operators are never left in doubt about what is going on.

6. Skilled Staff and so Acceptance of Long Type-ahead Sequences

The aim is to allow for the staff becoming expert in the use of the system and so learning to type ahead; the response must be fast enough to keep up with the typing or the buffer must be adequate for whenever the response is slower than that. This is not possible to do with Windows as mouse movements to get to the button to click require the button to be on screen.

The use of the mouse slows down even an experienced user such as myself. I have been using mice for eight or nine years and when I had to learn the key combinations necessary to operate Windows 95 without a mouse (my portable's pointing device failed) I was surprised how much quicker it was.

To build a system with the features listed above is much easier if the GUI can be ignored and everything done from the keyboard. All responses will be faster if the screen is character-based rather than a 640 by 480 or, worse, a 1024 by 768 graphic of pixels and it will be great deal easier to test such a system thoroughly as there are fewer ways of initiating action and fewer paths through it.

Many Windows systems are not adequately tested. On my portable without the pointing device there are many menus that do not properly allow for a keyboard alternative to the mouse click. For example the mouse set up procedure is impossible without a mouse because at one point the key "OK to proceed" didn't respond to <enter> even though it was marked as containing the focus!

There are other advantages to this approach. Every Window has to be described and takes space even if a table-based approach to Window definition such as Causeway is used. Working in APL*PLUS/PC version 8.1 the DOS address limit means that the interpreter and workspace both have to be contained in the 640K (slightly increased by the virtual workspace if you use it). The menu mechanism shown in the Appendix can be used for every menu in the system and it only takes 50 lines (it can be reduced to thirty or less by removing comments and more extensive use of the diamond separator). In the environment where such menus are used there is a file containing the tables of choices and the corresponding sets of help panels with the list of their names ready for use in the left argument. All these variables are commonly stored in a package so that the calling function can read them, fix them in the workspace, get the user's choice from the menu and expunge the package and its contents from the workspace, all in a few lines of code and use very little workspace indeed apart from the temporary variables.

Even old computers with slow chips such as 486 and even 386 run fast enough to support such systems.

Simpler systems are also easier to maintain.

Systems on this plan may be faster, more reliable, easier to write, and easier to use, require less training or maintenance and run on cheaper computers; these advantages are comparatively unimportant - the reason people choose them is that they suit their needs better.

Appendix - APL*PLUS/PC version 8.1 functions

There are six functions here; *QLX* is the top level that displays a menu of workspaces each of which will do everything required for one of the options. *BOFS* displays the menu, shows (*SHOWHELP*) and removes (*REMHELP*) the help and returns the rows that were selected. *BCHART* puts a bar chart on the screen. I haven't bothered with an example of how it is called. I hope the comments are helpful. The functions *MKCLRS* and *SETKEYS* in the *QLX* set up colours as delta-variables and action keys and their actions. *BOX* draws lines round a block of text.

```
    ∇ QLX;⎕IO;ΔA;ΔB;ΔE;ΔI;ΔH;ΔN;ΔS;PAK;PNAMES;OKEYS;Δ;WS
[1]   ⍝ ⎕LX function for the demonstration WS AJC 17 JAN 1999
[2]   ⍝ Model for top of menu structure of other QLXs
[3]   ⍝ Sets up colours (ΔA etc), keys, top menu, network globals, current date
[4]   ⍝ offers the menu of workspace options with this one highlighted
[5]    ⎕IO←0 ⋄ 48 ⎕POKE 161 ⋄ MKCLRS ⋄ 1 ⎕POKE 255 ⋄ ⎕TCFF
[6]    PAK←⎕FREAD 38 67 ⋄ PNAMES←ΔPNAMES PAK ⋄ ΔPDEF PAK ⋄ OKEYS←SETKEYS ' '
[7]   ⍝ 1-line per option menu is in 38 67
[8]   AGN:→(¯1 12 =WS←HELPS BOF WSS)/END,THISWS ⍝ WS is row in WSS or ¯1
[9]   ⍝ HELPS is a list of help panel names
[10]   ⎕QLOAD WNAMS[WS;] ⍝ Each choice is a WS: WSlist in WNAMS
[11]   ⋆ ⍝ After ⎕QLOAD next instr never obeyed
[12]  THISWS:RUN ⋄ →AGN
[13]  END:Δ←⎕EX PNAMES ⍝ PAK & PNAMES itself are localised
    ∇
    ∇ R←HELPS BOFS
TAB;⎕IO;LWIN;OKEYS;NOH;TRS;TCS;SR;NSP;SUL;SDL;OLDSCR;RWIN;ZWIN;Δ;WINDOW;OHS;
SCRN;DNS;UPS;TOP;HWS;OFFSET;SSR
[1]   ⍝ R←row from TAB (¯1 if esc). Simple menu AJC 8 JAN 1999
[2]   ⍝ TAB is a table of choices - (can be more than a screen)
[3]   ⍝ optional arg HELPS is NL of help screens, per row in TAB
[4]   ⍝ SR←→no of screen rows - can be varied as needed
[5]   ⍝ SUL←→value of TOP when at scroll-up limit
[6]   ⍝ SDL←→value of TOP when at scroll-down limit (=0)
[7]   ⍝ --------- 1 Set up
[8]    ⎕IO←0 ⋄ R←¯1 ⋄ OFFSET← 0 0 ⍝ top left of menu
[9]    NOH←2≠⎕NC 1 5 ⍴'HELPS' ⍝     NO Help
[10]   Δ← ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ¯1 ⎕WKEY enter,esc,ctrlend,tab,btab,
ucsr,dcsr,lcsr,rcsr ⋄ TOP←DNS←UPS←0 ⍝ vars are key values
[11]   TRS←1↑⍴TAB ⋄ TCS←¯1↑⍴TAB ⍝   TRS←→TAB rows  TCS←→TAB cols
[12]   SR←TRS⌊23-1↑OFFSET ⍝         SR←→Screen Rows (BOX adds 2)
```

```
[13]  SSR←(TRS,2)⍴⍺E,⍺A ⍝          Tally of selected rows
[14]  NSP←SR>1↑⍴TAB ⍝              NSP←→No Scrolling Poss 1∨0
[15]  SUL←0⌈(TRS)-SR ⋄ SDL←0 ⍝     Scroll-Up-Limit & Scroll-Down-Limit
[16]  LWIN←(⌈0.5×SR)⌊⌈0.5×TRS ⍝    Focus starts at middle
[17]  RWIN←(↓(4,SR)⍴(OFFSET[0]+1+⍳SR),(SR⍴2+OFFSET[1]),(SR⍴1),SR⍴TCS),(SR,4)⍴
0 256 ,⍺E,⍺A
[18]  WINDOW←OFFSET,(2+SR),4+TCS ⍝RWIN & WINDOW never change
[19] ⍝ ---------- 2 Construct display. RED←→RE-Display
[20] RED:SCRN←BOX TAB[TOP+⍳SR;] ⋄ RWIN[; 6 7]←SSR[TOP+⍳SR;]
[21]  OLDSCR←WINDOW ⎕WGET 3
[22]  WINDOW ⎕WPUT SCRN,[1.5]⎕AV[⍺H]
[23] DIS:→NOH/WE ⍝                 WE←→Windows Execute
[24]  OHS←(HWS←OFFSET+ 1 4 +LWIN,TCS)SHOWHELP HELPS[TOP+LWIN;] ⍝ OHS←→Old Help
Screen
[25] ⍝ ---------- 3 Windows Execute & select action
[26] WE:ZWIN←LWIN ⎕WIN RWIN
[27]  →NOH/SA ⍝                    SA←→Select Action
[28]  HWS REMHELP OHS ⍝            HWS←→Help Window Start (see[24])
[29] SA:→(ZWIN[1]=esc)/END2
[30]  →(ZWIN[1]=ctrlend)/END3
[31]  →(ZWIN[1]=enter)/END1
[32]  →(DNS←(ZWIN[1]=tab)∨(ZWIN[1]=dcsr)∨ZWIN[1]=rcsr)/DN ⍝ DowN Scroll
[33]  →(UPS←(ZWIN[1]=btab)∨(ZWIN[1]=ucsr)∨ZWIN[1]=lcsr)/UP ⍝ UP Scroll
[34]  →DIS
[35] ⍝ ---------- 4 Actions
[36] ⍝ DN←→scroll up or cursor down. UP←→scrl down or cursor up
[37] ⍝ END2←→esc - abort.          END1←→selection & in ZWIN[2]
[38] DN:UPS←0 ⋄ →(NSP∨TOP=SUL)/ML ⍝ Can't scroll so Move Lwin
[39]  TOP←TOP+1 ⋄ →RED ⍝           Scroll up 1 line & → RED
[40] UP:DNS←0 ⋄ →(NSP∨TOP=SDL)/ML ⍝ Can't scroll so Move Lwin
[41]  TOP←TOP-1 ⋄ →RED ⍝           Scroll down 1 line & → RED
[42] ML:LWIN←|SR|LWIN+DNS-UPS ⋄ →DIS ⍝Cycle focus in direction req'd
[43] END1:SSR[TOP+LWIN;]←,(2 2 ⍴⍺I,⍺S,⍺E,⍺A)[(SSR [TOP+LWIN;1]=⍺S);] ⍝ ⍺I,⍺S
are normal & ⍺E,⍺A highlight colours
[44] ⍝                          SR←→Selected Screen Row(s) marked
[45]  RWIN[; 6 7]←SSR[TOP+⍳SR;] ⍝  Highlight selected rows
[46]  →DIS ⍝       To exit on first selection COMMENT THIS OUT
[47] END3:R←(SSR[;1]=⍺S)/⍳TRS ⍝    Output selected rows
[48] END2:→(1≠TEST)/1+⎕LC ⋄ ⍞←1 DISMSG BOX 'TEST RESULT FROM BOF IS ',▼R
     ⍝ DISMSG displays a message and waits for enter
[49]  WINDOW ⎕WPUT OLDSCR
    ∇


    ∇ R←ROWCOL SHOWHELP H;XS;WINDOW;SR;SC
[1]  ⍝ Displays mat H for item that ends at ROW,COL 8 JAN 1999
[2]  ⍝ H is (semi-global) name of help panel
[3]  ⍝ R is the old contents of window
[4]  ⍝ ⎕IO must be zero (set by BOF or BOFS the calling fn)
[5]  ⍝ SR & SC could be added to the left arg if needed
[6]   H←⍴,H ⋄ SR←24 ⋄ SC←80⍝Get panel SR, SC←→Screen Rows/Cols
[7]   →(~TEST)/⎕LC+1 ⋄ ⍞←1 DISMSG BOX('ROWCOL←→',▼ROWCOL),'⍴H←→',▼⍴H
```

```
[8]   →(SC<ROWCOL[1]+¯1↑ρH)/TOOBIG ⋄ →(SR<1↑ρH)/TOOBIG
[9]  AL:→(SR<XS←(1↑ρH)+1↑ROWCOL)/ADJ ⍝  If too big →ADJ
[10]  WINDOW←ROWCOL,ρH
[11]  R←WINDOW ⎕WGET 4
[12]  WINDOW ⎕WPUT H,[1.5]⎕AV[ΔH] ⋄ →0
[13] ADJ:XS←XS-SR ⍝          XS is no of lines H must be moved up
[14]  ROWCOL←ROWCOL-XS,0 ⋄ →AL ⍝ Move window start up to fit
[15] TOOBIG:H←BOX 'Sorry help panel too big to fit' ⋄ →AL
     ∇


     ∇ ROWCOL REMHELP OHS;XS;WINDOW;SR;SC
[1]   ⍝ Removes help for item ending at ROW,COL 20 AUG 1998
[2]   ⍝ OHS is previous contents of help panel
[3]   ⍝ NB SR & SC must be set at same values as in SHOWHELP
[4]   ⍝ They could be added to the left argument if needed
[5]   ⍝ NB ⎕IO←→0, set by BOFS which calls this fn
[6]    SR←24 ⋄ SC←79 ⍝          SR, SC←→ Screen Rows/Cols
[7]    →(SR<XS←(1↑ρOHS)+1↑ROWCOL)/ADJ ⍝ If too big →ADJ
[8]   OK:WINDOW←ROWCOL,ρOHS
[9]    WINDOW ⎕WPUT OHS
[10]   →0
[11] ADJ:XS←XS-SR ⍝       XS is no of lines OHS must be moved up
[12]   ROWCOL←ROWCOL-XS,0 ⍝  Move window start up
[13]   →OK
     ∇


     ∇ RANGE BCHART SOF;BAR;CHARS;DISP;⎕IO
[1]   ⍝ Display of progress - 15 NOV 98
[2]   ⍝ RANGE is total run SOF is progress so far
[3]   ⍝ Fixed posn allows calling function to save and restore
[4]   ⍝ screen at the end of the task whose progress is shown.
[5]    ⎕IO←0
[6]    DISP←BOX 1 60 ρ⎕AV[176]
[7]    15 5 3 64 ⎕WPUT DISP,[1.5]⎕AV[ΔH]
[8]    BAR←(CHARS←⌈60×SOF÷RANGE)/⎕AV[219]
[9]    (16 7 1 ,CHARS)⎕WPUT((1,CHARS)ρBAR),[1.5]⎕AV[ΔA]
     ∇
```

Namespaces: New Gem From Old Roots

by Alexander Balako
Email: abalako @ infostroy.ru

Introduction

The most interesting object, since nested arrays were implemented, is (in my opinion) the Namespace object. It was implemented by Dyadic Systems Ltd. in Dyalog APL Version 6. I think that developers do not use the full power of this object for several reasons. In this article I want to describe some features of namespace programming technologies that can make easy-to-program utility functions using namespaces.

Using Complex Names

What do I call complex names? Look at the example :

```
      )ns Name1
#.Name1
      )ns Name1.Name2
#.Name1.Name2
      )ns Name3
#.Name3
      )ns Name4
#.Name4
      Name1.A←1
      Name1.Name2.A←2
      Name3.A←3
      Name4.A←4
```

Here we create 4 namespaces and 4 variables A with different values. A simple call to these variables is described below:

```
      #.Name1.A
1
      #.Name1.Name2.A
2
      #.Name3.A
3
      #.Name4.A
4
```

Nothing interesting there. Yes?

And now try to type in the session the next example:

```
      #.Name1.Name2.#.Name3.##.Name4.A    ⍝ (A)
```

What value did you get? Yes, this call is equal to the call:

```
      #.Name4.A ⍝ (B)
4
```

I call Line (A) a "Complex" call, line (B) a simple call.

Using Complex Calls

Using complex calls is illustrated by the examples below.

If you put all the functions to create GUI objects in the namespace *#.WUTIL* and try to create some of them from another namespace:

```
      )NS #.WUTIL
#.WUTIL
      )CS #.WUTIL
#.WUTIL
      ∇CreateForm[⎕]∇
[0]   CreateForm Name
[1]   Name ⎕WC'Form' 'Example'('Posn' 10 10)('Size' 50 60)
      )CS #
#
```

Here we initialise in namespace *#.WUTIL* the function *CreateForm* that will create a form at position 10 10 with size 50 60.

```
      )NS #.APP
#.APP
      )CS #.APP
#.APP
      #.WUTIL.CreateForm 'Form1'
```

and we call this function from namespace *#.APP*. But what happened: where was 'Form1' created?

```
      ⎕←⎕NL 9
```

Not in the current namespace. It is created in the namespace *#.WUTIL*.

```
      ⎕←#.WUTIL.⎕NL 9
Form1
```

Now change your function *CreateForm* as follows:

```
      )CS #.WUTIL
#.WUTIL
      ∇CreateForm[⎕]∇
[0]   CreateForm Name
[1]   Name←(⎕IO⊃⎕NSI),'.',Name ⍝ Take old namespace name and
add to Name
[2]   Name ⎕WC'Form' 'Example'('Posn' 10 10)('Size' 50 60)
```

And now try to call it:

```
      )CS #.APP
#.APP
      #.WUTIL.CreateForm 'Form1'
      ⎕←⎕NL 9
Form1
```

Line [1] of *CreateForm* changes the left argument from *'Form1'* to *'#.APP.Form1'*; if we call it with a left argument of *'##.APP.Form1'* this line changes it to *'#.APP.##.APP.Form1'*. So the form is relative to the current namespace.

Now using the complex name technique we can create a form where we want to create it. Even if we call the function from the same namespace:

```
)CS #.WUTIL
#.WUTIL
      CreateForm 'Form1'
      ⎕←⎕NL 9
Form1
```

Conclusion

I think that this little paper does not complete all the possibilities of using namespaces. Let's play with APL.

Windows BMP Files and J

by Brook Anthony Schofield
Email: Brook.Schofield @ utas.edu.au

Introduction

The purpose of this article is to describe my adventures with J in an attempt to create a script to run length encode (RLE) a Windows Device Independent Bitmap (DIB) file, which is more commonly known as a Bitmap or BMP.

The adventure began during a unit while studying for the final year of my degree. An exercise was proposed to explore functional programming and create a script that can functionally perform a task in the style used by the J programming language rather than the classical imperative programming that is usually taught. The task I selected, as described above, was to run length encode a bitmap image file.

Since J is designed along the same lines as APL as an array processing language I thought that a functional implementation of RLE would be relatively easy especially since a bitmap is described as "an image stored as an array of bits" by the Windows User Guide.

For avid readers of Vector you may notice that the title of this article is similar to one which appeared in Vector Volume 10, No.1, p.80 by Ray Cannon, entitled "Windows BMP Bitmap Files and APL".

While this article isn't a re-implementation of the topics covered in Ray's article it is based on a sentence from it:

> "Since compressing and expansion of bitmaps is best carried out in a compiled language, it is not discussed further."

While it may be best carried out in a compiled language, run length encoding is one of the simplest forms of compression and with the array processing features of the J programming language the tasks to implement this form of compression should be relatively simple.

This article starts with an outline of the structure of Microsoft Windows device independent bitmap files (BMP) and the format for run length encoding (RLE). This is followed by examples of J functions used for the manipulation of images in relation to the Windows Bitmap specifications for run length encoding.

Bitmaps

Microsoft Windows version 3.0 defined a standard for bitmap files and it has been in use ever since. There is also a bitmap "standard" for OS/2 machines, which is only slightly different from the Microsoft specification. These differences make the functions, described in this article, incompatible with the OS/2 format at the current time.

The Windows Device Independent Bitmap Format

Windows supports 4 different levels of colour in its Bitmaps:

- 1-bit, providing 2 colours which is often monochrome.
- 4-bit, providing 16 colours.
- 8-bit, providing 256 colours.
- 24-bit, providing 16.7 million colours.

Only 4 and 8-bit images allow compression using run length encoding and only 8-bit images have been used for the implementation of run length encoding within this article.

Both 4 and 8-bit encoding methods are described, with emphasis on the 8-bit method and how it relates to the J functions used for RLE image compression.

Bitmap data structures

Each Windows BMP file contains a file header, an information header, a colour table and bitmap bits; each of these are described below.

Offset	Size	Microsoft name	Description
0	2	bfType	The character BM for bitmap (66 77)
2	4	bfSize	The total size of the file
6	4	bfReserved1/2	Reserved, must be set to zero
10	4	bfOffBits	The offset to the bitmap bits from the beginning of the file.

Table-1: Windows Bitmap File Header

The purpose of the Windows Bitmap file header is to store the file type, size of the image and an offset to the start of the image data.

The Windows Bitmap file header doesn't need to be modified within the context of the Bitmap to RLE conversion, or vice-versa.

The `bfSize` double-word is based on the uncompressed size of the image, and since it is already set within the file it does not need to be modified, no matter whether the file is being compressed or decompressed.

The Windows Bitmap information header is modified within the manipulation functions because the information header contains a flag for the compression scheme. Although it is a 4-byte field there is only the option for uncompressed bitmaps or RLE images, represented by a 0 or 1 respectively, on the 16th bit.

Offset	Size	Microsoft name	Description
0	4	biSize	The size of this structure in bytes
4	4	biWidth	The width of the bitmap in pixels
8	4	biHeight	The height of the bitmap in pixels
12	2	biPlanes	Number of colour planes, set to 1
14	2	biBitCount	Colour bits per pixel. 1 4 8 or 24
16	4	biCompression	Code for the compression scheme
20	4	biSizeImage	Size of the bitmap bits in bytes
24	4	biXPelsPerMeter	Horizontal resolution in pixels per metre
28	4	biYPelsPerMeter	Vertical resolution in pixels per metre
32	4	biClrUsed	Number of colours defined in the pallet
36	4	biClImportant	Number of important colours in the image

Table-2: Windows Bitmap Information Header

The function below uses `toRLE` to manipulate the header to set the 30th bit to 1, *i.e.* the 16th bit of the information header after the 12-bit file header.

```
toRLE =: 1&}
head2rle =: (30 toRLE @(header {. nd@tl));header }. nd @ tl
```

The `head2rle` copies the header information to the left hand side (LHS) of the boxed values so they will not be altered by the encoding or decoding process as they are applied to the remaining boxed data. As stated, it sets the 30th element of the header (`biCompression`) to either 0 or 1 when converting from run length encoding or uncompressed bitmaps respectively.

The `biSize` is 40 for Windows DIBs and 12 for OS/2 DIBs, this causes an incompatibility between the current scripts if used with OS/2 bitmap files.

The colour table is used to specify the colours used within a bitmap image. This table is of no importance to the compression or decompression of bitmap images, but the integrity of the table has to be maintained. This is handled by the `header` function when called by the `head2bmp` or `head2rle` functions.

```
header =: 54 & [ + 4 & [ * 2 & [  ^ 28&{ @ nd @ tl
```

The header function calculates the offset to the bitmap data by determining the bit depth of the image and then bytes that the colour table consumes. For 8-bit images the colour table is always static, but this allows the scripts to be used with 4-bit images to make this function more general.

Offset	Size	Microsoft name	Description
0	1	rgbBlue	Blue intensity in the range 0-255
1	1	rgbGreen	Green intensity in the range 0-255
2	1	rgbRed	Red intensity in the range 0-255
3	1	rgbReserved	Set to 0

Table-3: Windows Bitmap Colour Table

OS/2 DIBs use an RGB triplet rather than the Windows 4-byte entry. This also causes a problem to the current scripts, if they where used on OS/2 bitmaps.

Run Length Encoding of Bitmap Files

Run Length Encoding (RLE) is only available for 4-bit and 8-bit images within the Windows BMP Specification. There are two forms of encoding within RLE:

- Encoded Mode
- Absolute Mode

Encoded Mode consists of two bytes, the first specifying the number of consecutive pixels to be drawn using the colour index which is specified by the second byte.

Absolute Mode is an escape sequence from Encoded Mode where the first byte is zero. The succeeding byte specifies one of four actions:

0 End of Line

1 End of Bitmap

2 A delta designated by two bytes which form unsigned values for the horizontal and vertical offsets to the next pixel from the current position. The area skipped is drawn in the background colour of the image.

3 Any value that is three or greater is taken as the number of bytes to be drawn verbatim, since there is no run of common pixels.

Compressed Data	Expanded Data
03 04	0 4 0
09 1E	1 E 1 E 1 E 1 E 1
05 06	0 6 0 6 0
00 00	End of Line
00 01	End of RLE bitmap
00 02 05 01	Move 5 right and 1 down
00 06 45 56 67 00	4 5 5 6 6 7

Table-4: 4-bit Run Length Encoded Data

The third (number 2) absolute mode option is not implemented within this article and the fourth option is only implemented for decompression, not encoding.

Below is an example of some bitmap data in compressed and expanded format. The data in Table-4 is for 4-bit bitmap image data. The main difference between 4 and 8-bit image data is that the latter is stored per byte, while the former is stored per nibble. A nibble being the high and low order 4-bits of a byte.

```
nibbles =: 16 16 & #:
```

The above function can be used to split the high and low order nibbles of a byte so that they can be treated separately. 4-bit images will not be the topic of any further discussion.

The data for 8-bit images, which is the format catered for by the script, allows each byte to represent either a pixel (uncompressed bitmap) or an encoded mode control which will ultimately represent a pixel.

Compressed Data	Expanded Data
03 04	04 04 04
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
05 06	06 06 06 06 06
00 00	End of Line
00 01	End of RLE bitmap
00 02 05 01	Move 5 right and 1 down
00 03 45 56 67 00	45 56 67

Table-5: 8-bit Run Length Encoded Data

Handling Files

File handling is performed using the `read` and `write` functions:

```
read =: 1!:1@<

boxed =: '' &; @ read

write =: 1!:2<

writing =: write~ "1 bd @ hd
```

These functions read in the data from a specified file name and box the data as explained in the *Iteration* section below. The *writing* of the data calls on a utility function to return a binary version of the data from numeric data.

Utility Functions

Various utility functions are used throughout the J script. The explanation given to each within the Appendix is sufficient for their operation, they will be explained further in their appropriate sections.

Iteration

Iteration is used extensively within the script to automate the encoding/decoding of arrays of data. This is accomplished by executing a function to the power of infinity. The stopping condition for this is achieved when the same result is returned twice by the function. This is demonstrated in the example below:

```
bmp2rle =: writing ]`do_encode@.(*@#@tl)^:_ @ head2rle @ boxed
```

The do_encode function is called via the use of a gerund (`). The gerund is triggered by returning the signum of the tally of the tail of the input data. When data is present in the right hand side (RHS) of the boxed data the function will be called repeatedly. Once the data on the RHS has become exhausted the gerund will call the] operator which will simply return the previous value and therefore halt the otherwise infinite loop. This style of iteration is used throughout the script.

	66 77 86 82 0 0 0 0 0 0 54 4 0 0 40 0 0 0 200 0 0 0 100 0 0 0 1 0 8 0 1 0 0 0 32 78

Table-6: Boxed Data Structure For Iteration

Above is a boxed version of the input data (being displayed in numerical data format rather than binary). The left hand side (LHS) box is initially empty. As data is transferred from one side of boxed list to the other, the function is re-called, with the result being further encoded data added to the LHS of the boxed data and fewer elements remaining in the RHS of the data.

Bitmap Compression

The functions used for the compression of the bitmap data match the elements in the data to be processed, up to a maximum of 255 matches, as that is the largest value that can be stored as an 8-byte value.

```
match =: E.~ {.

matches =: 255 &<. @ i. & 0 @ match
```

These matches are combined with the element that they are matching on, to form a run length encoded pixel representation.

...12 182	34 34 34 34 34 255 255 192 13 52 52 52 182 182 52 182 17 17 17 63 63 63...

Taking the example above, the boxed data contains some encoded data on the LHS and some uncompressed data on the right hand side (RHS). The matching process will return five instances of 34, so after the next iteration of the function the data will be as follows:

...12 182 5 34	255 255 192 13 52 52 52 182 182 52 182 17 17 17 63 63 63 63 63 63...

These functions are combined to create an *encode* function which is passed a boxed version of a single line of the bitmap image which is then iterated upon within the `inner_loop` function.

```
mh =: matches , {.

mt =: }.~ matches

mtotalhd =: (mh @ tl ,@:(,"1~) hd);mt @ tl

inner_loop =: (]`mtotalhd@.(*@#@tl) ^:_)
```

The `inner_loop` function is called from another iterative loop, on the `do_encode` function. The result of each iteration is a boxed version of the data, which is slightly modified through the matching of the data at the head of the RHS and then removing that data from the next instance of the boxed data.

```
convert_a_line =: 0 0&[ ,"1~ hd,"1 hd@inner_loop@boxw

remove_a_line =:  padding4 @ getw }. tl

do_encode =: convert_a_line ; remove_a_line
```

The `padding` utility function is used in the `remove_a_line` function to ensure that any padding material is removed from the RLE image. Bitmap images have to end their lines on 32-byte boundaries and therefore there can be up to 3-bytes of data that is unwanted from each line of an image. Padding data is generally random, taken from undefined memory locations.

Bitmap Decompression

The decoding of a RLE bitmap image is more time and memory consuming than the encoding of it. The reason being that while you know the width of the BMP image you don't know how many elements you will need to process until you have reached the end of a line. Remembering that bitmap images must have their lines end on a 32-byte boundary.

The `copy` functions below are used to remove elements from the RHS of the boxed data, usually in groups of two, and expand them to their original values.

```
lit =: ({.2&}.)~ {. @ }.

cpy =: {. # {. @ }.

copy =: lit`cpy@.(*@{.)
```

The cut functions have to remove the same number of elements from the RHS of the data, that are being copied into the expanded data box.

```
chop =: }.~ 2&[ + padding2 @ {. @ }.

trunc =: 2&}.

cut =: chop`trunc@.(*@{.)
```

The `chop` command is taking into account the *Absolute Mode* options for replication of uncompressed data and removing any padding data that may exist.

```
padit =: padbmp`empty@.((1&{ *@>. 0&{)@tl)

padbmp =: 1&[ #~ pad4 @ getw

empty =: ''&[
```

The added complexity of the `padit` function is to cater for the instance where End of Line (zero zero) is found so that the image can be padded out to the correct number of bytes.

```
do_decode =: (copy @ tl ,@:(,"1~) hd , padit); cut @ tl
```

The entire `do_decode` function is shown above, which encompasses all the described decoding functions.

Processing the Bitmap Image Information

In order to process and bitmap or run length encode bitmap files the following functions can be executed:

```
'output.bmp' bmp2rle 'input.bmp'
```

`bmp2rle` will process an uncompressed bitmap file in *input.bmp* and encode the data and output a valid run length encoded bmp image into *output.bmp*.

```
'output.bmp' rle2bmp 'input.bmp'
```

`rle2bmp` will process a compressed bitmap file in *input.bmp* and decode the data and output a valid bitmap image into *output.bmp*.

Notes on J

The code was written using J version 4.01a and uses some operators that do not work under older versions of J, such as version 3.

The code does not require you to run under Microsoft Windows™, in fact all testing was performed on a Macintosh™. This is why example file paths are incomplete, as they will need to be localised for your system type.

It is recommended that you have a bitmap image viewer/manipulator available if you wish to test the code and view the output files.

All the code was written functionally, and it is intended that you use the functions interactively for the processing of your images. Also note that the process can be time-consuming depending on your machine type and the size of the image that you are attempting to manipulate.

Conclusion

Microsoft Device Independent Bitmaps are relatively simple in format and can be easily converted from uncompressed to compressed versions for both 4 and 8 bit images which support compression.

As a further exercise to the reader, you may like to attempt the implementation of 4-bit RLE of bitmaps based on the principles described in this article, the author would love to hear of any attempts made. Also since the functions do not support OS/2 DIBs, modifications to the functions to support these, and possibly other file formats, would be a worth while exercise.

Acknowledgements

I would like to thank Ray Cannon and Adrian Smith for their assistance in providing me with a copy of the article "Windows BMP Files and APL" when the library resources that I had access to were lacking.

Thanks also to Mark Feldman from the PC Game Programmers Encyclopaedia (PC-GPE) for his information on the Windows BMP File format.

And to quote from Ray Cannon's article again "However, the reverse process, conversion from Bitmap to Metafile, is "not trivial". ("Not trivial" is a euphemism for "I don't know how do it"!)

Appendix - Code

```
NB. File Input Output
NB. read in data from a file and box it up
read =: 1!:1@<
boxed =: '' &; @ read
NB. write unboxed binary data to a file
write =: 1!:2<
writing =: write~ "1 bd @ hd

NB. Utility Functions
NB. get the head (first box) or tail (second box)
hd =: ,@ (>@{:)
tl =: ,@ (>@{:)
NB. returns 'numeric data' for comparison operations
nd =: a.&i.
NB. translates 'numeric data' back into binary data
bd =: { & a.
NB. get the width of the image from the 18th element of the header
getw =: 18&{ "1 @ hd
NB. get a boxed version of width elements of the data
boxw =: ''&[;getw {. tl
NB. pad the data out to two or four bytes for various functions
pad4 =: (0 3 2 1&[) {~ 4&|
padding4 =: + pad4
pad2 =: (0 1&[) {~ 2&|
padding2 =: + pad2
```

```
NB. Header File Processing
NB. The first 54 bytes are BMP header information followed by a colour table
NB. the colour tables size is:    4 * 2^bit depth
header =: 54 & [ + 4 & [ * 2 & [  ^ 28&{ @ nd @ tl
NB. change the header information so that the image is a RLE or BMP image
toRLE =: 1&}
head2rle =: (30 toRLE @(header {. Nd@tl));header }. nd @ tl
toBMP =: 0&}
head2bmp =: (30 toBMP @ (header {. Nd@tl));header }. nd @ tl

NB. RLE8 -> BMP8
NB. function for processing a file from a RLE encoded image to a BMP image.
rle2bmp =: writing ]`do_decode@.(*@#@tl)^:_ @ head2bmp @ boxed

NB. decode the RLE bit patterns which are either literal or copied
lit =: ({.2&}.)~ {. @ }.
cpy =: {. # {. @ }.
copy =: lit`cpy@.(*@{.)
NB. cut the decoded lines from the data to be processed
chop =: }.~ 2&[ + padding2 @ {. @ }.
trunc =: 2&}.
cut =: chop`trunc@.(*@{.)
NB. place padding bits for 32bit word at the end of the line for a BMP
padit =: padbmp`empty@.((1&{ *@>. 0&{)@tl)
padbmp =: 1&[ #~ pad4 @ getw
empty =: ''&[
NB. encapsulate the processing so that it can be iterated upon
do_decode =: (copy @ tl ,@:(,"1~) hd , padit); cut @ tl

NB. BMP8 -> RLE8
NB. function for processing a file from a BMP image to a RLE encoded image.
bmp2rle =: writing ]`do_encode@.(*@#@tl)^:_ @ head2rle @ boxed
NB. determine the value to match on for a run of data, maximum 255 matches
match =: E.~ {.
matches =: 255 &<. @ i. & 0 @ match
NB. the number of matches at the head and the value matched on
mh =: matches , {.
NB. drop the number of matches from the tail/input data
mt =: }.~ matches
NB. the match head total should be boxed the same as the presented data for
iteration
mtotalhd =: (mh @ tl ,@:(,"1~) hd);mt @ tl
inner_loop =: (]`mtotalhd@.(*@#@tl) ^:_)
NB. return the new head of the box
convert_a_line =: 0 0&[ ,"1~ hd,"1 hd@inner_loop@boxw
NB. return the new tail of the box
remove_a_line =:  padding4 @ getw }. tl
NB. encapsulate the processing so that it can be iterated upon
do_encode =: convert_a_line ; remove_a_line
```

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: A Special Locator in Dyalog APL/W	Joachim Hoffmann	108
Technical Correspondence		110
At Play with J: New Big Deal	Gene McDonnell	113
Purging Name Pollution in APL+Win	Ajay Askoolum	120
Exact Horadam Numbers with a Chebyshevish Accent	Clifford A. Reiter	122
J/LAPACK Interface...	Procter & Brown	133

Hacker's Corner: A Special Locator in DyalogAPL/W

by Joachim Hoffmann (joho@ping.at)

Imagine you have to implement a special locator functionality inside a subform, where the locator should change its cursor to indicate if it is over a valid or invalid target object. So the user should see where it makes sense to drop his locator. Also you want to draw a special locator line, *e.g.* a dogleg instead of a straight line.

I solved this problem with some tricky ⎕*NA* calls to the Windows API. On Objects where the locator is started a MouseDown handler returns 0 in order to interrupt the default mouse processing on the start-object. The same callback also sets the mouse capture via User32|SetCapture on the parent form. This has the effect that from now on all mouse events are processed by the parent form. This is where the MouseMove handler of the parent form starts to do its work. It checks on each call if the mouse is waved over a valid target object and sets the appropriate cursor on the parent form. On the MouseUp of the parent form the MouseMove handler is deactivated and the appropriate action is taken depending on where the locator was dropped, or in an even more object-oriented design an EndLocator event could be fired.

And as its MOST IMPORTANT action *it releases the capture* back to default processing (user32|ReleaseCapture) so that the other objects can receive mouse events again. I also set a global ⎕*TRAP*, so that in case of error the capture would be released.

```
      ∇ InitQuadNACalls
[1]     ⎕NA'I user32.C32|WindowFromPoint {I I}'
[2]     ⎕NA'I user32.C32|ChildWindowFromPoint I {I I}'
[3]     ⎕NA'I user32.C32|GetCursorPos >{I I}'
[4]     ⎕NA'I user32.C32|ScreenToClient I ={I I}'
[5]
[6]     ⎕NA'I user32.C32|SetCapture I'
[7]     ⎕NA'I user32.C32|ReleaseCapture'
      ∇
```

In the MouseMove handler the object under the cursor needs to be detected. As a prerequisite I have registered the handles of all valid target objects in a global variable at creation time (obj ⎕*WG* 'handle' after ⎕*WC*). The MouseMove handler

first queries the cursor position (user32|GetCursorPos), which is then translated from screen coords to client coords of the parent form (user32|ScreenToClient).

```
      ∇ xy←MousePosQna hdl;r
[1]     (r xy)←GetCursorPos 2 ⍝just a dummy argument
[2]     (r xy)←ScreenToClient hdl xy
      ∇
```

The position is then fed in turn into user32|ChildWindowFromPoint (CWFP), in order to query if there is a child window under the cursor. Please beware that CWFP also returns deactivated and/or invisible child windows, which have to be taken care of. (While writing this article I just found a new service called ChildWindowFromPointEx, for which you can now specify whether it should omit hidden or disabled child windows - at least it says so in MSDN Lib April 99). So if CWFP returns a handle, it's a quick job to check it against the list of valid targets. I had to use CWFP instead of the simpler version WindowFromPoint (WFP), because WFP only returned the handle of the APL Session object.

```
     ∇ formhdl OnMouseMove msg;ps2;chdl;target_type
[1]   ⍝∇ MouseMove-handler on parent form
[2]    ps2←MousePosQna formhdl  ⍝ x/y(!) in parent form
[3]    chdl←ChildWindowFromPoint formhdl ps2
[4]    :If ~0∈⍴chdl  ⍝ if there is a child window
[5]        DrawSpecialLocator Δps1,[0.5]⌽ps2
[6]        target_type←formhdl CheckTarget chdl
[7]        SetCursor target_type ⍝ set Cursor prop on parent form
[8]    :Else ⍝ no handle → outside of parent
[9]        SetForbiddenCursor 1
[10]   :End
     ∇
```

Using this technique I could implement quite an efficient locator with hit-testing. This could also be used for a variety of special drag&drop operations.

TECHNICAL CORRESPONDENCE

From: Joseph De Kerf 24 May 1999

In a recent paper [1], J. Sullivan extends in a very elegant way the domain of the subscript n of the Fibonacci numbers F_n from positive integers to negative integers, reals, and complex numbers. Just one remark.

Referring to my paper [2], he quite rightly notices that the function *FIB1* published in my paper, using only the first term of the golden section formula, doesn't work for negative values of n, the function giving zeros:

```
      ∇ F←FIB1 N;R
[1]   F←⌊0.5+(÷R)×(0.5×1+R←5*0.5)*N
      ∇

      +N←¯7+⍳13
¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5 6

      N,[0.5] FIB1 N
¯6 ¯5 ¯4 ¯3 ¯2 ¯1 0 1 2 3 4 5 6
 0  0  0  0  0  0 0 1 1 2 3 5 8
```

He states that "the values for negative n are the same in absolute value as those for positive n, but the signs alternate" and "in order to generate values for negative n as well as positive n you will need both terms of the golden section formula", giving for that purpose the function *FIB2*:

```
      ∇ F←FIB2 N;R
[1]   F←⌊0.5+(÷R)×((0.5×1+R)*N)-(0.5×1-R←5*0.5)*N
      ∇

      N,[0.5] FIB1 N
¯6  ¯5  ¯4  ¯3  ¯2  ¯1 0 1 2 3 4 5 6
¯8   5  ¯3   2  ¯1   1 0 1 1 2 3 5 8
```

I don't agree. The fact that the values for negative n are the same in absolute value as those for positive n, but with alternating signs, may be formalized by the relation $F_{-n}=-(-1)^n F_n$.

This means that, using this relation we may calculate F_{-n} from F_n by giving its value the appropriate sign, while calculating F_n only needs the first term. Incorporating the relation in *FIB*1, we get for instance the function *FIB*3:

```
     ∇ F←FIB3 N;R
[1]   F←(¯1*(N+1)×(N<0))×⌊0.5+(÷R)×(0.5×1+R←5*0.5)*|N
     ∇

      N,[0.5] FIB1 N
¯6  ¯5  ¯4  ¯3  ¯2  ¯1  0  1  2  3  4  5  6
¯8   5  ¯3   2  ¯1   1  0  1  1  2  3  5  8
```

References

[1] *J. Sullivan: Functional Extensions to the Fibonacci Sequence.* Vector Vol.15 No.4 pp.123-126

[2] *J. De Kerf: From Fibonacci to Horadam.* Vector Vol.15 No.3 pp.122-130

From: Peter Donnelly May 24 1999

"Knowledge, Age and Experience are no Substitute for Good Fortune"

or

"How, in the end, Bill wins — you lose"

I now know the cause of my embarrassing failure last Friday [at the AGM].

Given time, I had intended to close my talk with a demonstration that showed how a Dyalog APL ActiveX control (and DYALOG.DLL, if necessary) gets downloaded and installed by Internet Explorer if it is not already present on the user's PC. When this occurs, the packaged ActiveX control file (DUAL.CAB) is downloaded and decompressed, then DUAL.OCX is copied into the Windows\System directory and registered.

On Friday morning, having tested both mechanisms for installing DUAL.OCX (saving from Dyalog APL and downloading from the Internet), I de-installed the DUAL.OCX in Windows\System (using the standard Microsoft tool REGSVR32.EXE) and erased DUAL.OCX from the Windows\System directory. I then re-saved and re-registered DUAL.OCX (in d:\dyalog82) from Dyalog APL.

Being knowledgeable, elderly and experienced, and knowing Bill's propensity for fun at my expense, I verified that everything was registered correctly using OLEVIEW.EXE and I checked that DUAL worked in Visual Basic. Happy as a sandboy, I then packed up my laptop and left for the gig.

As we know, during my talk that afternoon to the BAA, my proudly presented DUAL.OCX failed to load in IE4, leaving egg on my face and undisguised mirth at the back of the room.

The explanation is *simple* ...

When IE4 downloads and installs an ActiveX control from the Internet, it records information keyed by its ClassID in (wait for it ...)

> HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Code Store Database\
> Distribution Units\

It is *this* information that IE4 subsequently uses when it sees an ActiveX control in a web page, conveniently ignoring the official ActiveX registration data.

I had installed DUAL.OCX from the Internet that very morning (as a test) then completely removed/de-installed it before re-installing directly from APL. An ActiveX control is responsible for creating/removing the entries that are officially required for ActiveX registration, and Dyalog APL fully conforms in this respect. However, it is a little unreasonable to expect APL to remove undocumented entries created by recalcitrant Microsoft products. As usual, Bill seems to be ignoring his own rules.

My DUAL.OCX demo failed because IE4 was looking for DUAL.OCX in C:\WINDOWS\SYSTEM, using the registry entries that IE4 had itself previously recorded in the Code Store Database section. In these circumstances, it seems that IE4 blithely ignores the official documented registry entries that are required by the ActiveX protocol, which in this case correctly pointed to C:\DYALOG82\DUAL.OCX.

At Play With J: New Big Deal

by Eugene McDonnell, with major contribution from Roger Hui

Chris Burke was displeased, to begin with. He had tried J's deal with a small left argument and a very large right argument, and this took a perceptibly long time. He tried it with a larger argument and was told he had run out of space. He tried it with a still larger argument, and was told he had exceeded deal's limits. He wrote to Roger Hui about these distressing circumstances, showing him several examples:

```
   (time=:6!:2) '1 ? 1e7'  NB. this takes too long
0.93

   1 ? 1e8
|out of memory
| 1 ?100000000

   1 ? 1e9
|limit error
| 1 ?1000000000
```

Roger's first thought was how to get around this limitation. He said that if the left argument is much smaller than the right Chris would be better off just doing `?x$y` ; because deal allocates a boolean vector of length y to compute unique answers in the result of `x?y`. Thus a very large y would give the results Chris noted. He forwarded Chris's message to me, copying Chris, with the message "Perhaps Eugene can comment."

I checked the literature on my shelf but couldn't find any worthwhile "selection without replacement" algorithms. I then remembered the very early days of APL\360, before deal was made a primitive — some time in 1966, perhaps. I had written a defined function to perform deal, using an algorithm that was quite fast. I remembered it vaguely, and thought that with it I might be able to do better with J. I wrote one, tried it out on a dozen or so cases, then communicated it to Chris and Roger:

```
deal =: dyad define
NB. experimental deal
NB. for small x. and very large y.
count =: 0
t =. i.0
k =. 1.1 NB. adjust as you see fit
NB. maybe make it a function of y.
```

```
u =. >. k * x.
whilst. (# t) < x. do.
t =. ~. ? u # y.
count =: count + 1
end.
x. {. t
)
```

As you can see, I was uncertain about the factor 1.1. I had put a counter in to see how often more than one execution of the whilst section was needed. In the few dozen cases I tried, there were none. Happily, the timings showed a large improvement over current deal for Chris's cases: the cases that were slow were much faster, and the range of the right argument was significantly extended. Here are the timings I experienced:

```
   time '1 ? 1e7'
4.07
   100 time '1 deal 1e7'
0
   1000 time '1 deal 1e7'
0.00044

   1000 time '1 deal 1e8'
0.00044

   1000 time '1 deal 1e9'
0.00044

   1000 time '100 deal 1e7'
0.00082

   count
1
   1000 time '100 deal 1e8'
0.00076
   count
1
   1000 time '100 deal 1e9'
0.00083
   count
1

   ts=: 6!:2 , 7!:2@] NB. time and space

   100 ts '1?1e7'
0 2240
   100 ts '1?1e8'
0 2240
   100 ts '1?1e9'
0.0005 2240
```

```
   100 ts '100?1e7'
0.0005 4288
   100 ts '100?1e8'
0.0005 4288
   100 ts '100?1e9'
0.0005 4288
```

Roger thought this was neat. He implemented my algorithm, invoked when `x<0.01*y`. He rewrote my algorithm, simplifying it:

```
deal =: dyad define
u =: >. 1.1 * x.
while. x.># t=. ~. ? u # y. do. end.
x. {. t
)
```

His C implementation looked like this:

```
static A bigdeal(m,n)I m,n;{A t,x,y;
 RZ(x=sc((I)floor(1.11*m)));
 RZ(y=sc(n));
 do{RZ(t=nub(roll(reshape(x,y))));}while(m>AN(t));
 R vec(INT,m,AV(t));
} /* E.E. McDonnell circa 1966, small m and large n */
```

But he worried that this would run into the birthday problem, which gets its name from its most celebrated instance, that the odds are in your favour if you bet that in a group of 23 or more people at least two of them will have the same birthday. The greater the number of people, of course, the higher the probability that this will occur. With more than 365 people, the probability is certain, or 1, since the pigeon hole principle dictates that if you have x pigeonholes and y pigeons, with x less than y, at least one of the pigeonholes will have more than one pigeon in it. What Roger wanted to know was, what is the value of x as a function of y where the probability of a duplicate appearing was 0.5. If he had this information, he could make the multiplier more accurate. Perhaps 1.11 wasn't good enough.

I wrote the following:

```
Hui_test =: dyad define
tests =: 0
successes =: 0
whilst. 1000 > tests =. >: tests do.
successes =: successes + */~: ?x.$y.
end.
<. 100 * 0.001 * successes
)
```

I ran this test for y in all of the hundreds, thousands, ten thousands, hundred thousands, millions, and 10,000,000 and 20,000,000. To make it easier to digest I'll only show the results for `y=: 10^2 3 4 5 6 7`. The other results are consistent with these.

```
        y     x
      100    12
     1000    37
    10000   116
   100000   370
  1000000  1180
 10000000  3740
```

I thought this looked roughly like a square root relationship so tried a few manoeuvres:

```
   %: y
10 31.6228 100 316.228 1000 3162.28
   %: 1.4*y
11.8322 37.4166 118.322 374.166 1183.22 3741.66
```

This was quite a good fit, and shows:

```
y = (x^2) % 1.4
x = %: 1.4 * y
```

to a close approximation. I communicated these results to Roger. He studied this and was able to apply some more theory to it, and wrote back to me: in choosing `m` items from a universe of `n` items, the probability of all the `m` items distinct is `(*/n-i.m) % (n^m)`. The numerator is the number of ways of choosing `m` distinct items; the denominator is the number of ways of choosing `m` items. Thus:

```
(*/n-i.m) % (n^m)                              (a)
(*/n-i.m) % (*/m$n)
*/ (n-i.m) % (m$n)
*/ (n-i.m) % n

f=: [: */ ] %~ i.@[ -- ]

   22 23 f"0 ] 365 NB. the classic birthday problem
0.524305 0.492703
```

The first `m` for which `m f n` is less than 0.5, is:

```
1 + (0.5 > */\ n %~ n - i.n) i. 1
```

```
   f1=: >: @ (i.&1) @ (0.5&>) @ (*/\) @ ([ %- [ - i.)
   f1 365
23

   n=: , (,10^1 2 3 4 5)*/1 2 4 8
   m=: f1"0 n
   n,.m ,. %: 1.4*n
    10     5 3.74166
    20     6  5.2915
    40     8 7.48331
    80    11  10.583
   100    13 11.8322
   200    17 16.7332
   400    24 23.6643
   800    34 33.4664
  1000    38 37.4166
  2000    53  52.915
  4000    75 74.8331
  8000   106  105.83
 10000   119 118.322
 20000   167 167.332
 40000   236 236.643
 80000   334 334.664
100000   373 374.166
200000   527  529.15
400000   745 748.331
800000  1054  1058.3
```

For extremely large values of `n`, Roger saw that the function `f1` is impractical to compute, and he concluded that a method based on root finding on the original function `f`, using my approximation of `m=: %: 1.4*n` as an initial guess, would be more suitable.

```
1e7 3724
1e8 11775
1e9 37234
2e9 52656
```

The end result was that Roger found that the rule I had originally suggested of switching to the "roll" algorithm for `m?n` when `m<0.01*n` does run into the birthday problem, and he replaced it with the more accurate rule I had found following his suggestion. There the matter rests, with one small postscript. As I studied Roger's mathematics, particularly the phrase `(*/n-i.m)` in line (a) above, I recalled J's "falling factorial" function. The J Dictionary defines this as follows:

> The fit conjunction applies to ^ to yield a stope defined as follows: `x!.k n` is `*/x + k*i. n`. In particular, `^!._1` is the falling factorial function.

Let me elaborate on this. Think of the caret (^) as being defined in the first instance as denoting a function of three arguments: a base, a count, and a step. Then caret (base, count, step) is the product over base + step * integers count. For example,

```
'base count step' =. 3 5 7
   i. count
0 1 2 3 4
   step * i. count
0 7 14 21 28
   base + step * i. count
3 10 17 24 31
   */ base + step * i. count
379440

caret =: monad define
NB. general caret function
NB. y. is a list of three values:
NB.  base
NB.  count
NB.  step
'base count factor' =. y.
*/ base + factor * i. step
)
   caret(3 5 7)
379440
```

This generality hides the fact that there are really just three variations of significant interest, steps having values `_1`, `0`, and `1`. These three provide falling factorials, steady factorials (powers) and rising factorials. Each of these has to do with a product over count number of values, beginning with base, and continuing with values increasing by step. See what results come with a base of 5 and a count of 3, with each of the three significant step sizes:

```
   caret 5 3 _1
60
   5 * 4 * 3
60
   caret 5 3 0
125
   5 * 5 * 5
125
   5 ^ 3
125
   caret 5 3 1
210
   5 * 6 * 7
210
```

The case with step zero is the default case of caret, and is the power function. We can use the falling and steady factorial cases to write a more compact version of Hui's function f:

```
probdupes =: dyad define
NB. Probability of duplicates when drawing with
NB. replacement from among the first count integers
base =. x.
count =. y.
(base ^!._1 count) % (base ^!.0 count)
)

NB. simplified version of probdupe
pd =: ^!._1 % ^

NB. version exploiting family resemblances
pdx =: [: %/ ^!._1 0

   365 probdupes 23
0.492703
   365 pd 23
0.492703
   365 pdx 23
0.492703
```

These functions fail when the values of the falling factorial get very large, causing its value to exceed the maximum real value, and thus to be represented by machine infinity. When this happens, the steady factorial (power) value will already be machine infinity, and the quotient of two infinities is *Not a Number*, or NaN. To get around this problem, use extended integers as arguments, and extended precision inverse on the result. When the result is smaller than the smallest nonzero number, a result of zero is forced:

```
   365 pd 200                    NB. result is NaN, from _ % _
_.
   x:^:_1 [ 365x pd 200x         NB. result of zero is forced
0
```

Purging Name Pollution in APL+Win

by Ajay Askoolum

Variables in APL are global by default, but can be localised. *{It may be preferable if it were the other way round.}*

Imagine a Windows front end managing a legacy back end. Legacy systems tend to be badly written, and badly documented.

How does the front end clear the debris left by the legacy system?

It is usually not practical to track all the variables created in the legacy system. However, it is easy to identify the one function which starts the legacy system – it is the one called by the front end.

1. Insert the word ProcPurge as the first statement on the first executable line in this function
2. Add a line with the word Purge at the end, ensuring that this function exits via the last line.
3. Localise Purge

What are ProcPurge and Purge?

```
    ∇ ProcPurge
[1]   ⍝ Ajay Askoolum
[2]   0 0⍴⍎def ('∇Purge',(,';',⎕nl 2)~' '),⎕tcnl,'[1]0 0⍴⎕ex ⎕nl 2∇'
    ∇
```

ProcPurge creates a function Purge which localises all variables in the workspace prior to calling the legacy system. Running the function Purge as the last thing before returning to the front end wipes out all variables created by the legacy system.

If the front end relies on the variables created by the legacy system, initialise the required variables prior to calling the legacy system. These variables will be modified but will not be deleted.

By the same token, the legacy system can modify any variable that existed before it started.

How does the front prevent the legacy system from modifying its variables?

Modify the ProcPurge function as shown below:

```
    ∇ ProcPurge
[1]   ⍝ Ajay Askoolum
[2]   ⍎⎕def ('∇Purge',(,';',⎕nl 2)~' '),⎕tcnl,'[1]0 0⍴⎕ex ⎕nl 2 ⋄ Legacy ∇'
    ∇
```

Note the second statement on the first line of Purge, Legacy. This is the function which starts the legacy system.

Thus, ProcPurge starts the legacy system, protecting all variables in the workspace. Be sure to run Purge when returning back to the front end.

Note

The above method will work for functions too (`⎕nl 3`) and for both variables and functions at the same time. Take care *not* to localise the function stated in ProcPurge.

The catch? The length of the names of variables or functions (or both) separated by ; and including the string ProcPurge must not exceed the maximum length of 32768 bytes.

Try it!

Exact Horadam Numbers with a Chebyshevish Accent

by Clifford A. Reiter (reiterc@lafayette.edu)

A recent paper by Joseph De Kerf illustrated the use of Binet type formulas for computing Horadam numbers [1]. This was an effective demonstration of the power of those formulas and the grace of APL for implementation. That article spurred my interest in the Horadam numbers - a long neglected interest. One of my earlier related interests was in computing very large Fibonacci numbers [6]. That work takes advantage of identities involving Fibonacci numbers but does not directly use the Binet formula. Since the Binet formulas are most naturally implemented with floating point computations, the computations are rapid, but of finite precision, on most systems. Thus, my goal in this note is to describe a very different way of computing Horadam numbers and to implement those ideas in J where exact rational computations are available. Moreover, we will see several additional examples of the pervasive nature of the Horadam numbers including a glance at Chebyshev polynomials. The recursive algorithm that we will use also illustrates a powerful J programming style. While matrix constructions of the Fibonacci numbers are well known [2], the corresponding matrix constructions for Horadam numbers are not nearly as well known, but they do appear [9].

Matrix View of Horadam Numbers.

Following the notation in [1], we will define the Horadam sequence by

$$H_1 = p, H_2 = q, \quad H_{n+1} = sH_n + rH_{n-1}.$$

However, the notation for these numbers varies greatly. For example, Horadam [3] and Rabinowitz [5] use a notation where the recursion contains a minus sign and in many contexts it would be more convenient to begin the sequences with H_0. Notice that H_n as defined above depends upon p, q, r, and s in addition to n. When we need to emphasize some or all of that dependency, we can write ${}^{rs}_{pq}H_n$ or ${}_{pq}H_n$ for H_n.

We rewrite the above recursion in matrix form as

$$\begin{pmatrix} H_n \\ H_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ r & s \end{pmatrix} \begin{pmatrix} H_{n-1} \\ H_n \end{pmatrix}.$$

Iterating this n-1 times, we see

$$\begin{pmatrix} H_n \\ H_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ r & s \end{pmatrix}^{n-1} \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}.$$

Considering the initial condition, $H_1 = p = 1, H_2 = q = 0$, we see

$$\begin{pmatrix} {}_{10}H_n \\ {}_{10}H_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ r & s \end{pmatrix}^{n-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Likewise, when $H_1 = p = 0, H_2 = q = 1$, we see

$$\begin{pmatrix} {}_{01}H_n \\ {}_{01}H_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ r & s \end{pmatrix}^{n-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Putting these together, we have

$$\begin{pmatrix} {}_{10}H_n & {}_{01}H_n \\ {}_{10}H_{n+1} & {}_{01}H_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ r & s \end{pmatrix}^{n-1}.$$

Since ${}_{pq}H_n = p({}_{10}H_n) + q({}_{01}H_n)$, we can compute general Horadam numbers by computing powers of the matrix $W = \begin{pmatrix} 0 & 1 \\ r & s \end{pmatrix}$. In particular, ${}_{pq}H_n$ is the dot product of the vector $< p, q >$ with the first row of W^{n-1}.

Moreover, the matrix identities $W^{mn} = \left(W^n\right)^m$ and $W^{m+n} = W^m W^n$ give rise to powerful identities that allow one to skip many intermediate steps in computing Horadam numbers. Indeed, in the Fibonacci case, identities arising from those matrix relations give methods for rapid computation of very large Fibonacci numbers [6]. The identities for the Horadam numbers may be written explicitly and implemented (a worthwhile diversion - see Appendix B). However, we will use the graceful array capabilities of J to utilize the matrix relations directly. In particular, we will recursively compute even powers of W by $W^{2n} = \left(W^n\right)^2$ and compute the odd powers of W using $W^{2n+1} = W\,W^{2n}$. In the next section we will implement these ideas.

Horadam Numbers in J

We first define functions that give the 0th and 1st powers of W. In all the functions we define for giving powers of W, we will let the left argument be the list r,s and the right argument be the ultimately desired power – even though some of that information is unused for these low powers.

```
   w0=: (=i.2)"_
   2 3 w0 50                    0th power of W
1 0
0 1
   w1=: 0 1&,:@x:@[
   ]W=.2 3 w1 50                1th power of W
0 1
2 3
   x=: +/ . *                   Matrix product

   W x W                        W^2
2  3
6 11
```

Next we consider the general cases for computing non-negative powers of W. There are 4 cases according to whether the power is 0, 1, a general even power, or a general odd power. The function `case` distinguishes between these cases by adding the power mod 2 to twice the result of the Boolean test: "is the power greater than 1?".

```
   case=: 2&| + +:@(1&<)
   (,:case)i.8                  We see how the four cases depend on the power
0 1 2 3 4 5 6 7
0 1 2 3 2 3 2 3

   we=:[ x-@:w -:@]             Even powers of W: the square of the half power

   wo=:w1 x [ w <:@]            Odd powers of W: W times the previous power

   w=: w0`w1`we`wo@.(case@])    Agenda on the four cases gives general powers

   2 3 w 2                                  W^2
2  3
6 11
   2 3 w 50                                 W^50
 52015835828755613305113714 2  926285732032534439103474303
185257146406506887820694860 6 329901555438515945036156005 1
```

Recall that the Horadam numbers are the dot product of the list p,q with the first row of W^{n-1}. We will follow [1] and take the left argument of our Horadam function to be the list p,q,r,s. Thus `2&{.@[` gives p,q and `_2&{.@[` gives r,s.

The definition of `horadam` may be read as *p,q* dot product with the first row of W^{n-1}. We apply rank 1 0 since we expect the left argument to be a vector and the right argument to be the number of the desired term in the sequence.

```
   horadam=: (2&{.@[ x _2&{.@[ {.@w <:@])"1 0

   1 0 2 3 horadam 51
52015835826755613305113714 2
```

The following exact integers are consistent with the correct entries in the table in the appendix of [1].

```
   ,.2 3 _3 3 horadam 249 250 251
_1455578342930689280434675661902780082182495258305659396184 81
                                                              0
 4366735028792067841304026985708340246547485774916978188554 43

   sxfmt 2 3 _3 3 horadam 249 250 251    Short exact format
_145557834...(60)...5939618481
0          ...(1)...          0
4366735028...(60)...7818855443
```

The function `sxfmt` is a utility for displaying the beginning and end of exact integers. It is convenient to use it when the number of digits is very large. These "short exact format" expressions also include the number of decimal digits of the exact integer in parentheses. The definitions of `sxfmt` (short exact format) and a related function, `sxmfmt`, (short exact matrix format) are given in Appendix A and are available on-line [7].

We can also use the `horadam` function for non-integer *p,q,r,s* and maintain exact computations since exact rational arithmetic is built-into J. The experiments below are again consistent with the correct entries in the appropriate table in [1].

```
   x: _0.6 0.98 _2.7225 _3.3              Change decimal to exact rational
_3r5 49r50 _1089r400 _33r10
   float=: x:^:_1
   float x: _0.6 0.98 _2.7225 _3.3        Change back to decimal
_0.6 0.98 _2.7225 _3.3
   float (x: _0.6 0.98 _2.7225 _3.30) horadam 100 101 102
0 3.39555e19 _1.12053e20
```

Some Special Horadam Sequences

De Kerf [1] noted some special cases of Horadam numbers and other special cases are noted in [3,4]. We consider these to demonstrate the pervasiveness of the Horadam numbers.

```
   1 3 _1 2 horadam >:i.10
1 3 5 7 9 11 13 15 17 19
```
Arthimetic progression starting 1 3

```
   3 8 _1 2 horadam >:i.10
3 8 13 18 23 28 33 38 43 48
```
Arthimetic progression starting 3 8

```
   1 3 0 3 horadam >:i.10
1 3 9 27 81 243 729 2187 6561 19683
```
Geometric progression

```
   1 1 1 1 horadam >:i.10
1 1 2 3 5 8 13 21 34 55
```
Fibonacci numbers

```
   1 3 1 1 horadam >:i.10
1 3 4 7 11 18 29 47 76 123
```
Lucas numbers

```
   1 2 1 2 horadam >:i.10
1 2 5 12 29 70 169 408 985 2378
```
Pell numbers

```
   1 3 1 2 horadam >:i.10
1 3 7 17 41 99 239 577 1393 3363
```
Pell-Lucas numbers

```
   1 1 2 1 horadam >:i.10
1 1 3 5 11 21 43 85 171 341
```
Jacobsthal numbers

```
   1 5 2 1 horadam >:i.10
1 5 7 17 31 65 127 257 511 1025
```
Jacobsthal-Lucas numbers

In fact, because of the two-step recursive definition of the Horadam sequence, many other classical sequences and functions may be obtained from the Horadam numbers. For example, the Chebyshev Polynomials [8] are recursively defined by

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \text{ where } T_0(x) = 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1.$$

Thus, the value of $T_n(x)$ is the n^{th} Horadam number with p,q,r,s equal to $x, 2x^2-1, -1, 2x$.

```
   chebyshev=: (, <:@+:@*: , _1: , +:)@]"0 horadam [
```

```
   2 chebyshev 0.9
0.62
```
$T_2(0.9)$

```
load 'plot'
   $Y=: 1 2 3 4 5 chebyshev"0 1 X=: _1++:(i.%<:)100
5 100
   plot X;Y
```

Figure 1 shows the resulting plot of five Chebyshev polynomials. As one descends just to the left of x=1, the 1st through the 5th Chebyshev polynomials are crossed in order. Do you notice properties of these polynomials? Several can be observed.

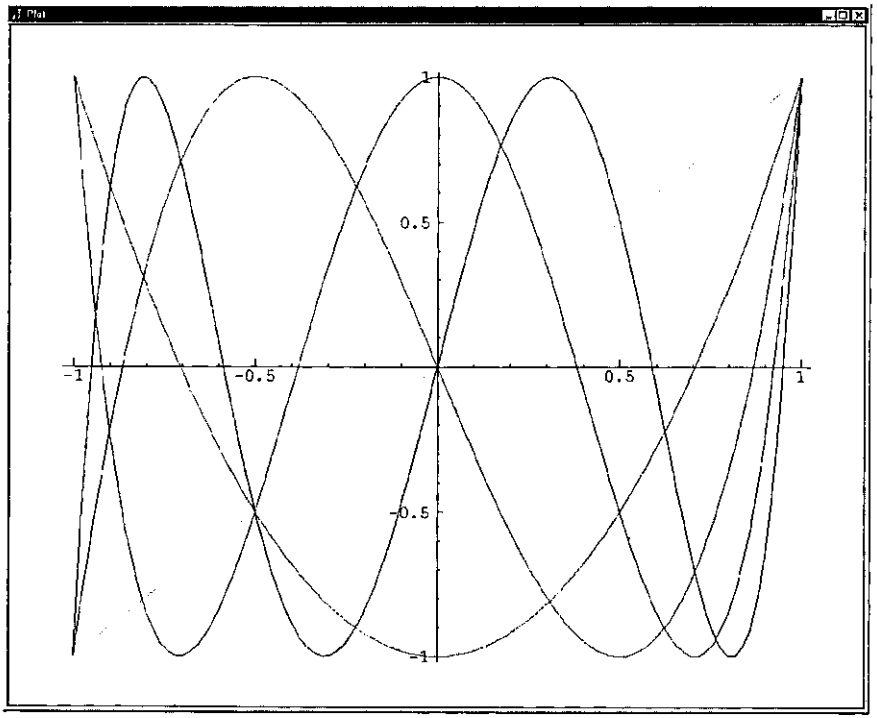


Figure 1. Five Chebyshev Polynomials

Timing and Efficiency

The efficiency of this matrix approach to computing Horadam numbers is of interest. First, we can temporarily modify our function w so that we can see which powers were recursively called.

In particular, we change w as follows.

```
   wrsc=: (1!:2)&2                          Write to screen function

   w=: w0`w1`we`wo@.(case@wrsc@])           Modified w

   1 1 1 1 horadam 100
99
98
49
48
24
12
6
3
2
1
354224848179261915075
```

Since there can never be two or more odd powers in a row, no more than $2\log_2(n)$ recursive steps are required. Our approach could be improved in a number of ways. We only used the top row of the highest power of W that we compute; so we ought to be able to speed things up by not computing the bottom row. Some improvement can also be expected if we use $W^{2n+1} = (W\,W^n)W^n$ in order to compute the odd powers of W. However, more dramatic improvements (between a factor of 2 and 4) can be obtained by implementing vector/scalar formulas instead of using matrices. See Appendix B for a brief description and implementation.

Table 1 shows the required computer time to compute various size Fibonacci numbers using `horadam` on a 400MHz Pentium based computer. The 5 milliseconds needed to compute exactly the 209 digit Fibonacci number F_{1000} is not as fast as the floating point computations done in [1], but even accounting for the differences in processor speed, is remarkably close and very respectable. We see that computations in J with thousands or tens of thousands of digits are routine while calculations producing a couple of million digits take significant amounts of time. We did not directly measure the space required for these computations, but the J session as a whole required less than 80M of memory.

n	F_{10^n}	time (sec)
1	55	0.001125
2	354224848179261915075	0.002374
3	4346655768...(209)...6849228875	0.005
4	3364476487...(2090)...9947366875	0.0438
5	2597406934...(20899)...3428746875	3.86
6	1953282128...(208988)...8242546875	398.172
7	1129834378...(2089877)...6380546875	62125.9

Table 1. Time required to compute some Fibonacci Numbers.

Appendix A. Definition Summary

We give here a summary of our J functions. Readers may obtain the script horadam.ijs from http://www.lafayette.edu/~reiterc/j/index.html [7].

```
x=:+/ . *
case=: 2&| + +:@(1&<)
w0=:(=i.2)"_
w1=:0 1&,:@x:@[
we=:[ x~@:w -:@]
wo=:w1 x [ w <:@]
w=: w0`w1`we`wo@.(case@])
horadam=:(2&{.@[ x _2&{.@[ {.@w <:@])"1 0
chebyshev=: (, <:@+:@*: , _1: , +:)@]"0 horadam [

wrsc=:(1!:2)&2
sxfmt=:(10&{.,'...('"_,":@(#-'_'&=@{.),')...'"_ , _10&{.)@":"0
sxmfmt=: ([,'    '"_ ,])/"2 @:sxfmt
```

Appendix B. Direct Implementation Using Identities

In squaring a 2 by 2 matrix we ordinarily compute 8 products. We will see the special nature of the Horadam numbers is such that we can do this with 3 products. In practice, this leads to almost a 8r3 fold improvement in efficiency. We will not give detailed derivations of the requisite identities, but will explain briefly how they may be obtained.

For convenience, we define

$$K_0 = 0, K_1 = 1, \quad K_{n+1} = sK_n + rK_{n-1}.$$

Thus, $K_{-1} = 1/r$ is a reasonable extension and in general $K_{n-1}={}_{01}H_n$ and it isn't too hard to show by induction on n that $rK_{n-2}={}_{10}H_n$. Therefore ${}_{pq}H_n = prK_{n-2} + qK_{n-1}$

We can verify that

$$W^n = \begin{pmatrix} rK_{n-1} & K_n \\ rK_n & sK_n + rK_{n-1} \end{pmatrix}$$

using the matrix power expansion of W in terms of the H's that we saw and then translating its entries into K's using the above relations.

Squaring the right side of $W^{2n} = \left(W^n\right)^2$ and equating the entries in the first rows yields:

$$rK_{2n-1} = r^2K_{n-1}^2 + rK_n^2 \text{ and } K_{2n} = 2rK_{n-1}K_n + sK_n^2$$

and hence

$$K_{2n-1} = rK_{n-1}^2 + K_n^2 \text{ and } K_{2n} = K_n(2rK_{n-1} + sK_n)$$

which goes from the pair (K_{2n-1}, K_{2n}) to (K_{n-1}, K_n) using just three "big" multiplications. We implement this as follows.

```
horadamb=:4 : 0"1 0
'p q r s'=.x: x.
k0=.((%r),0)"_
k1=.0 1x"_
ke=.((r,1)&x@:*: , {: * ((+:r),s)&x)@:k@-:
ko=.(0 1,:r,s)&x@k@<:
k=.k0`k1`ke`ko@.case
((p*r),q) x k <:y.
)
```

Lastly, we remark that $\det(W) = -r$, and hence

$$(-r)^n = \det(W^n) = rK_{n-1}(sK_n + rK_{n-1}) - rK_n^2.$$

That identity can be used to modify the formula for K_{2n-1} into a formula using just one "big" multiplication if one assumes that $(-r)^{n-1}$ is not a "big" computation relative to K_n. Then (K_{2n-1}, K_{2n}) may be computed from (K_{n-1}, K_n) using just two "big" multiplications. We leave the implementation of this approach as an exercise.

References

[1] Joseph De Kerf, From Fibonacci to Horadam, *Vector* 15 3 (1999) 122-130.

[2] Verner E. Hoggart, Jr., *Fibonacci and Lucas Numbers*, The Fibonacci Association, Santa Clara, 1969.

[3] A. F. Horadam, Special Properties of the Sequence $W_n(a,b;p,q)$, *The Fibonacci Quarterly*, 5 5 (1967) 424-434.

[4] A. F. Horadam, Jacobsthal Representation Numbers, *The Fibonacci Quarterly*, 34 1 (1996) 40-54.

[5] Stanley Rabinowitz, Algorithmic Manipulation of Second Order Linear Recurrences, *The Fibonacci Quarterly*, 37 2 (1999) 162-177.

[6] Clifford A. Reiter, Fast Fibonacci Numbers, *The Mathematica Journal*, 2 3 (1992) 58-60.

[7] Clifford A. Reiter, Horadam Numbers, http://www.lafayette.edu/~reiterc/j/index.html, 1999.

[8] Theodore J. Rivlin, *The Chebyshev Polynomials*, John Wiley & Sons, New York, 1974.

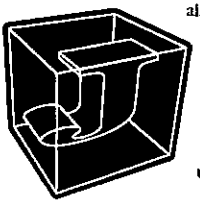
[9] Marcellus E. Waddill, Matrices and Generalized Fibonacci Sequences, *The Fibonacci Quarterly*, 12 4 (1974) 381-386.

Clifford A. Reiter
Department of Mathematics
Lafayette College
Easton, PA 18042 USA

IT'S SURPRISING
HOW JSOFTWARE
MAKES YOUR DATA

JUMP OUT!

J for Windows 95/NT from Iverson Software Inc. provides everything you need to solve problems quickly. You can build complete stand alone Windows applications, or easily integrate J with other programs. For example, use J as an ActiveX control for a web page, an OLE server for Excel, or a 32-bit DLL called from a C program. J is also available on a variety of other platforms including Windows CE, Mac, Linux, Solaris and AIX. The language is identical in all versions, and programs not making use of platform-specific features will work unchanged on all systems.



J in a Box

Strand Software 19235 Covington Court Shorewood, MN 55331
Tel: (612) 470-7345 Fax: (612) 470-9202
www.jsoftware.com

J/LAPACK Interface Makes Econometric Analysis Fly!

Richard J. Procter (rjp@interlog.com) and
Richard L.W. Brown (rlwbrown@YorkU.CA)

Introduction

In November 1998, Iverson Software Inc. announced J Release 4.02 and presented a demonstration of the product to the Toronto APL Special Interest Group. J4.02 contains new features including memory-mapped files and the focus of this discussion, the J/LAPACK interface. Iverson Software notes that the latest release of J, for Windows 9x/NT, Mac and WinCE, is available at their website at <www.jsoftware.com>. Recently, another J "addon" product was made available: the FFTW library - a collection of fast C routines for computing the Discrete Fourier Transform in one or more dimensions.

The J/LAPACK interface provides a specialized and sophisticated addition to the software toolkit of the array languages programmer. The following is from the release notes at the Iverson Software website:

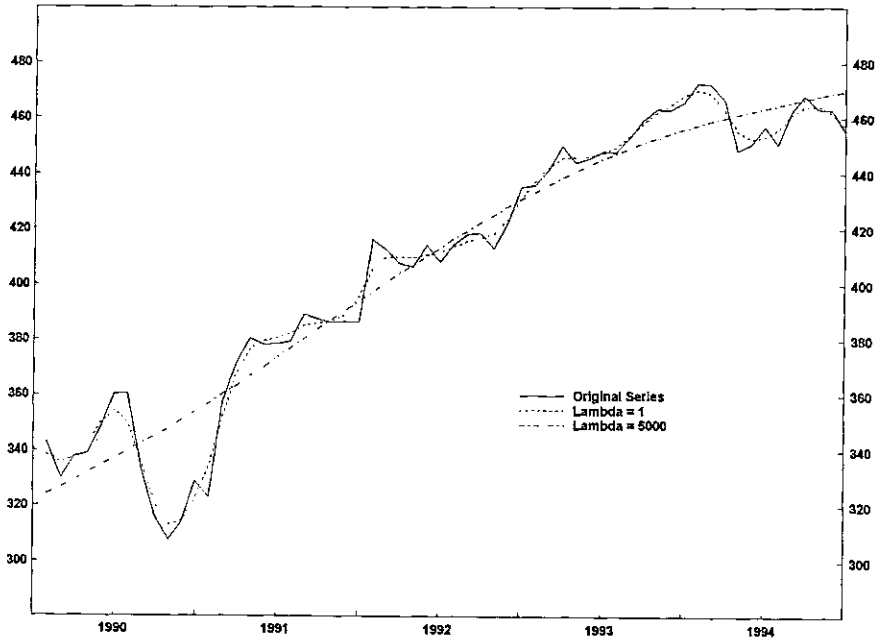
> "The complete LAPACK library is now available with the J LAPACK AddOn. LAPACK (Linear Algebra Package) is a set of routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers."

In a recent analysis of financial and economic time series information, the J/LAPACK interface was used to greatly enhance the speed of computations for a PC/WindowsNT system written in Dyalog APL. The purpose of the exercise was to optimize the smoothing parameter for a business cycle analysis function known as the Hodrick-Prescott filter. In this case, using the J/LAPACK interface represents a dramatic example of incorporating a specialized and optimized code library to solve a major computational problem in an existing system.

The Hodrick-Prescott Filter

The HP filter is similar to other methods for determining trends in time series data such as moving averages, in that it generates, from the given series, a

calculated trend series which may be compared with the original data to determine potential buy (or sell) decision signals. In theory, if the original data series rises above (or falls below) the calculated trend, a buy (or sell) signal is indicated. One input parameter to the HP filter, lambda, determines the degree of smoothing of the original data that is incorporated into the trend. Chart 1 (below) indicates the concept of HP filter and original series crossing points, and the effect of lambda on the HP filter outcome.



Optimum Return: the Right Lambda

Viewed graphically, as the lambda value is increased, variation in the HP filter result is reduced and the outcome less closely resembles the original time series. Typically, this means fewer crossing points between the original and calculated series occur. By applying the generated buy/sell signals at these crossing points to existing historical data, we can determine a net return value, based on the percent increase or decrease between transactions. To calculate the optimum return, we recalculate the HP filter curves over a large range of lambda values (perhaps 100 or more values) and compare the returns.

To complete the analysis, optimum lambda is calculated for an initial portion of the timeframe of the chosen series, say the first 5 years of a 15-year daily frequency time series. When optimum lambda is calculated for this portion and applied to the series, we calculate a final buy or sell signal for that segment, based on the relative positions of the original series and the HP filter at the last data value. At this point, we "roll forward" by one or more values in the time series, and repeat the process. This roll forward of the "calculation window" continues until the end of the series. From these successive buy/sell signals, we can calculate an overall return, used to determine whether applying the HP filter analysis at optimum lambda will be of use for predicting future directions.

At the client site, this optimization process was part of a larger economic analysis system written in Dyalog APL running in a PC environment under Windows NT. The HP filter function in APL was initially coded by converting an existing COBOL routine, complete with scalar looping algorithm, into APL. Although useful for calculating the HP filter for one time series, this strategy was found to take anywhere from 30 minutes to several hours of computation, even on the fastest PC, when applied to the full set of calculations for the optimization of lambda as described above.

The HP Filter Function - Array Solution

Let $y=[y_t]_{t=1}^{N}$ be a series of numbers representing financial data. It is desired to filter out the noise and compute a trend series $x=[x_t]_{t=1}^{N}$ that smooths the given series. The Hodrick-Prescott filter [1,2] is defined to be the solution to the minimization problem:

$$\min_{[x_t]_{t=1}^{N}}\left[\sum_{t=1}^{N}(y_t-x_t)^2+\lambda\sum_{t=2}^{N-1}(\Delta^2 x_t)^2\right]$$

where $\Delta^2 x_t=(x_{t+1}-x_t)-(x_t-x_{t-1})$ is the second difference (the discrete version of the second derivative). The parameter $\lambda>0$ is the smoothing parameter. If λ is small then the trend series x is very close to the given series y because the first summation above is dominant. If λ is very large then the second summation is dominant and the trend series is very smooth (i.e. has a very small second difference). To solve the minimization problem, set the partial derivatives with respect to each x_t (for $t=1 \ldots N$) equal to zero.

This gives N equations in N unknowns x_1 to x_N. In matrix form, the equations are: $y = (I + \lambda K)x$ where I is the identity matrix and K is the matrix:

$$\begin{bmatrix}
1 & -2 & 1 & 0 & \dots & & & & & 0 \\
-2 & 5 & -4 & 1 & 0 & \dots & & & & 0 \\
1 & -4 & 6 & -4 & 1 & 0 & \dots & & & 0 \\
0 & 1 & -4 & 6 & -4 & 1 & 0 & \dots & & 0 \\
0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 & \dots & 0 \\
\dots & & & & & & & & & \dots \\
\dots & & & & & & & & & \dots \\
\dots & & & & & & & & & \dots \\
0 & \dots & & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\
0 & \dots & & & 0 & 1 & -4 & 6 & -4 & 1 \\
0 & \dots & & & & 0 & 1 & -4 & 5 & -2 \\
0 & \dots & & & & & 0 & 1 & -2 & 1
\end{bmatrix}$$

We solve for the trend x as follows: $x = (I + \lambda K)^{-1}y$.

In APL, we could use a looping iterative algorithm to solve the system but this approach, as mentioned above, proved to be too slow. We could use quad-divide to solve the linear system. However, quad-divide is a general purpose linear solver and our linear system has special properties that can be exploited: the coefficient matrix is symmetric and banded with just two bands above and below the diagonal. LAPACK has specialized routines for solving this type of problem. Appendix A shows how the appropriate LAPACK routine is called from J. It also gives some benchmark comparisons of several solutions which were investigated using J, including J's %. (*i.e.* J's version of quad-divide), the LAPACK general solver (i.e. LAPACK's quad-divide), and the LAPACK symmetric banded solver.

Benchmarks

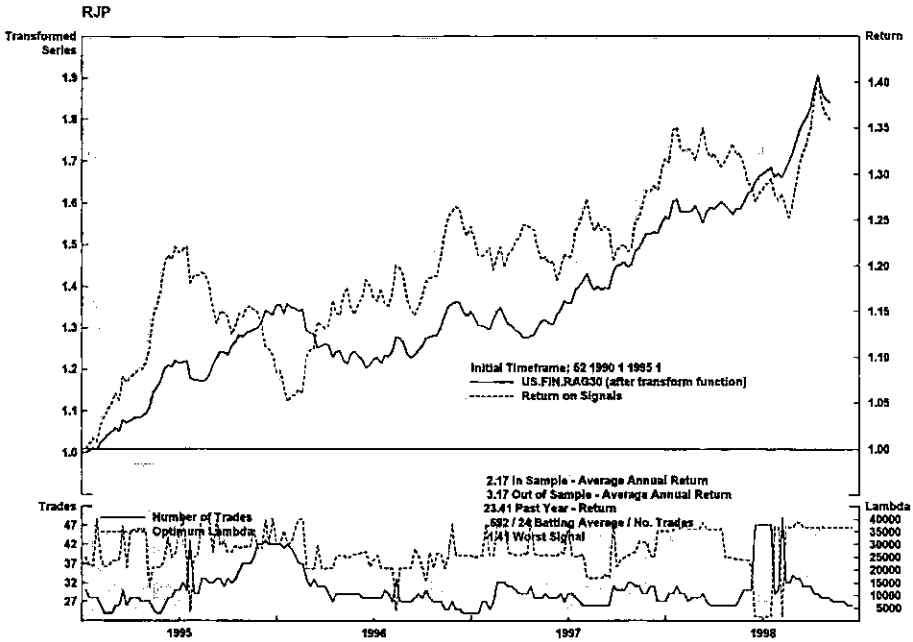
As an example of benchmark comparisons for the entire HP filter analysis, comparing APL itself without using J, vs. using the J/LAPACK interface, the following results are indicative.

For an analysis of weekly bond price information, from 1990 to the end of 1998, a 5-year calculation window was used, for 194 different lambda values (ranging from 10 to 40000) in each iteration. An iteration means that the calculation

window is rolled forward, in this case, by a 3-week interval, starting with 1990 to 1995, week 1, and ending at the last week of 1998.

Using APL only, and the looping HP filter algorithm which the system was originally coded in, this analysis took about 83 minutes to complete, running on a Pentium-II PC at 200 Mhz. The same analysis, using the J/LAPACK interface, and the `solve3` verb (symmetric banded solver defined above), took only 2 minutes, an impressive 40-fold increase in speed.

This sort of speedup was generally observed for all cases tested so far. In conclusion, since the analysis could now be done with a realistic turnaround time, economists were able to complete their investigations on whether the HP filter is a suitable tool for predicting future trends, by applying the calculations over various timeframes and datasets. Chart 2 shows a typical analytical result, indicating buy/sell periods (shading) and other useful information.



References

1. Hodrick, R. and Prescott, E., *Post-War U.S. Business Cycles: An Empirical Investigation*, Carnegie-Mellon working paper, 1980.

2. Reeves, J.J., Blyth, C.A., Triggs, C.M., and Small, J.P., *The Hodrick-Prescott Filter, a Generalisation, and a New Procedure for Extracting an Empirical Cycle from a Series*, Working Paper No. 160, August 1996, Department of Economics, the University of Auckland, New Zealand. (Available on the web at http://stat.auckland.ac.nz/reports/report96/oths9602.html)

Appendix A - J Script for LAPACK Interface

At the time of writing, the J-interface for LAPACK's symmetric positive definite band solve routine was not included in the routines supplied with the J/LAPACK addon product. Hence, we added that interface as an extra file named 'DPBSV.ijs' in the folder 'j402\system\packages\lapack'. This interface file, given below, was easy to develop using the documentation supplied in the J/LAPACK addon and especially by studying the other interface routines that are supplied.

START OF FILE: DPBSV.ijs

```
coclass 'jlapack'
NB. ======================================================
NB.*dpbsv v solves the real system A * X = B
NB. where A is an N by N symmetric positive definite banded matrix.
NB.
NB. form: dpbsv mata;matb
NB.
NB. returns: X
NB.
NB. The matrix A is assumed to be banded with KD super and sub
NB. diagonal bands.  For example, if N=6 and KD=2 then
NB.
NB.       a11  a12  a13   0    0    0
NB.       a12  a22  a23  a24   0    0
NB. A =   a13  a23  a33  a34  a35   0
NB.        0   a24  a34  a44  a45  a46
NB.        0    0   a35  a45  a55  a56
NB.        0    0    0   a46  a56  a66
NB.
NB. But A is stored and given to dpbsv in the sparse form
NB.
NB.        *    *   a13  a24  a35  a46    (* = anything)
NB. A =    *   a12  a23  a34  a45  a56
NB.       a11  a22  a33  a44  a55  a66
dpbsv=: 3 : 0
'ma mb'=: y.
if. (iscomplex ma) +. iscomplex mb do.
  need 'zpbsv'
```

```
  zpbsv y.
  return.
end.
if. 2 ~: #$mb do. mb =.((#mb),1)$,mb end.
n=. 1{$ma
if. n ~: #mb do.
  'dpbsv' error 'matrices should have same number of rows'
  return.
end.
uplo=.'U'
nrhs=. {:$mb
ldab=. #ma
kd=. ldab-1
ldb=. n
ab=. dzero + |:ma
b=. dzero + |:mb
info=. izero
arg=. 'uplo;n;kd;nrhs;ab;ldab;b;ldb;info'
(cutarg arg)=. 'dpbsv' call ".arg
if. info~:0 do.
  'dpbsv' error 'info result: ',":info return.
end.
if. nrhs=1 do. b else. |: (nrhs,n)$ b end.
)
```

END OF FILE: DPBSV.ijs

To benchmark the routine, we must define some sample data in J. The J-benchmark times in this Appendix were done under Windows 95 on a Pentium 75 computer with 16 megabytes of memory.

START OF FILE: hpfilter.ijs

```
n =: 200                    NB. Number of entries in time series
y =: ?n#30                  NB. Random time series
K =: (n,n){.(n,n+1)$6 _4 1,((n-3)#0),1 _4  NB. Start making K
J1 =: ,{,&<~ 0 1            NB. Indices for modifying K
J2 =: ,{,&<~ (n-1),(n-2)    NB. Indices for modifying K
K =: 1 _2 _2 5 1 _2 _2 5 (J1,J2)} K         NB. Modify K
SK =: 1,(0 _2,((n-3)#_4),_2),:(1 5,((n-4)#6),5 1) NB. Band version of K
NB. SK = band version of K. It has three rows as follows:
NB.        [ 1   1   1   1   1 . . .  1   1   1   1  ]
NB.  SK =  [ 0  _2  _4  _4  _4 . . . _4  _4  _4  _2  ]
NB.        [ 1   5   6   6   6 . . .  6   6   5   1  ]
lambda =: 1600
A =: (=i.n) + lambda*K          NB. A is (1 + lambda*K)
SA =: (0,0,:n#1) + lambda*SK    NB. Band version of A
time =: 10&(6!:2)               NB. Average time in seconds for 10 runs
require 'system\packages\lapack\lapack.ijs'
require 'system\packages\lapack\dgesv.ijs'
require 'system\packages\lapack\dpbsv.ijs'
solve1 =: %.-                    NB. J's 'dyadic domino'
```

```
solve2 =: ,@dgesv_jlapack_@;,.   NB. LAPACK's generic solve
solve3 =: dpbsv_jlapack_@;       NB. LAPACK's symmetric band solve
```

END OF FILE: hpfilter.ijs

```
NB. Benchmarks:
   time 'A solve1 y'
10
   time 'A solve2 y'
0.4
   time 'SA solve 3 y'
0.016
```

Appendix B – APL Functions Used in the HP Filter Analysis

The Total Return Optimization (TRO) system is written in Dyalog APL and runs under Windows NT on PC hardware. The J/LAPACK interface is implemented to provide the rapid calculation of the HP filter on a moving window along a time series data vector for a wide range of smoothing parameter (lambda) values.

From APL, we first initialize the J calculation engine by starting the J "EXE Server", with the *jstart* function, which also loads the script *'hpfilter.ijs'*. This script creates several J verbs which are used in subsequent calculations. The *jopen* and *jcmd* functions, plus several other standard J-interface APL functions were supplied by Chris Burke of Strand Software Inc. in Toronto.

```
     ∇ jstart path;j
[1]   ⍝ rjp: start JEXEServer, run hpfilter.ijs script
[2]   ⍝ path = path to J scripts, with trailing \
[3]
[4]    j←jopen 0                          ⍝ start J
[5]
[6]    j←J.Do'load ''',path,'hpfilter.ijs'''
[7]    'J Verbs:'
[8]    jcmd'nl 3'
     ∇
```

The *jsolve3* APL function passes a pre-calculated vector of lambda values (smoothing parameter, e.g. 10 to 40010 in increments of 200) and the current time series data segment to the J/LAPACK interface, to calculate a set of HP filter series in array fashion. *jsolve3* creates a 3-dimensional array of banded symmetric matrices SA containing the necessary permutations of each lambda value for this approach. The `solve3` J verb is then applied to each sub-array of SA using the `each` verb and the time series segment JDATA.

```
     ∇ r←lam jsolve3 series;n;j
[1]   ⍝ rjp: HPfilter via J/LAPACK dpbsv.isj and solve3 verb
[2]   ⍝ hpfilter scripts by Richard L.W. Brown, York University
[3]   ⍝ J/array version: form arrays in J, execute LAPACK in one pass
[4]   ⍝ lam = vector of lambda values
[5]   ⍝ series = timeseries vector
```

```
[6]
[7]     series←,series
[8]     n←⍴,series
[9]
[10]    j←J.Set'JDATA'series
[11]    j←J.Set'LAM'lam
[12]    j←J.Do'n=. ',⍕n
[13]    j←J.Do'SK =: 1,(0 _2,((n-3)#_4),_2),:1 5,((n-4)#6),5 1'
[14]    j←J.Do'SA =: (<0,0,:n#1)+each LAM*each <SK'
[15]    r←jcmd'SA solve3 each <JDATA'
[16]    r←↑r
      ∇
```

Richard J. Procter
Software Consultant
Toronto, Canada
rjp@interlog.com

Richard L.W. Brown
York University
Toronto, Canada
rlwbrown@YorkU.CA

Index to Advertisers

Dyadic Systems	6
Mackay Kinloch	24
Strand Software	132
Soliton	2
APL Booklist	16
Vector Back Numbers	4

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.

British APL Association: Membership Form

Membership is open to anyone interested in APL. The membership year normally runs from 1st May to 30th April, but new members may join from 1st August, November or February if preferred. 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

(please tick the required option above)

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:

British APL Association, c/o Rowena Small, 8 Cardigan Road, LONDON E3 5HU, UK

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.

1999/2000 Committee

Chairman	Anthony Camacho 0117-973 0036 100612.1057@compuserve.com	11 Auburn Road, Redland, BRISTOL, BS6 6LS
Secretary	Ajay Askoolum 01737-771643 106173.3347@compuserve.com	42 Hanworth Road, Redhill, Surrey RH1 5HT
Treasurer	Nicholas Small 0181-980 7870 treas.apl@bcs.org.uk	8 Cardigan Road, LONDON E3 5HU
Journal Editor	Stefano Lanzavecchia stf@adaytum.dk	c/o Vector Administration, Brook House, Gilling East YORK YO62 4JJ
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 01388-605544 100021.3073@compuserve.com	Unit R21, Auckland New Business Centre, St. Helen Auckland, Bishop Auckland, Co. Durham DL14 9TX
Administration	Rowena Small 0181-980 7870 treas.apl@bcs.org.uk	8 Cardigan Road, LONDON E3 5HU

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), Richard and Adam Weber (0121-3546550), Anthony & Sylvia Camacho, Ray Cannon (01252-874697), Marc Griffiths, Bob Hoekstra (01483-771028), Jon Sandles (01904-612882)

Typeset by APL-385 with MS Word 5.0 and GoScript

Printed in England by Short-Run Press Ltd, Exeter

VECTOR

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

SUSTAINING MEMBERS

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

Causeway Graphical Systems Ltd
The Maltings, Castlegate,
MALTON, North Yorks YO17 7DP
Tel: 01653-696760
Fax: 01653-697719
Email: causeway@csi.com
Web: www.causeway.co.uk

Compass Ltd
Compass House
60 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: 0171-353 8900
Fax: 0171-353 3325
Email: 100020.2632@compuserve.com

Insight Systems ApS
Nordre Strandvej 119C
DK-3150 Hellebæk
Denmark
Tel: +45 49 76 20 20
Fax: +45 49 76 20 30
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