VECTOR

FOCUS ON NESTED ARRAYS

The Journal of the British APL Association

A Specialist Group of the British Computer Society

Vol.2 No.1 July 1985

CONTRIBUTIONS

All contributions to VECTOR should be sent to the Editor at the address given on the inside back cover. Letters and articles are welcomed 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 submitted in duplicate and accompanied by as much visual material as possible, including a photograph of the author. Unless otherwise specified each item will be considered for publication as a personal statement by its author, who accepts legal responsibility that its publication is not restricted by copyright. The provision of camera-ready or machine-readable copy is encouraged: please contact the Editor beforehand. Program listings should indicate the computer system on which they have been run. APL symbols should be displayed on a separate line and not embedded in narrative. Except where indicated, items published in VECTOR may be freely reprinted with appropriate acknowledgement.

MEMBERSHIP

Category	*Fee p.a.*	*VECTOR copies*	*Passes*
Nonvoting student membership	£ 5	1	1
UK Private membership	£ 9	1	1
Overseas private membership	£ 16	1	1
Supplement for airmail (not needed for Europe)	£ 8		
Corporate membership	£ 75	10	5
Sustaining membership	£325	neg	5

The membership year runs from 1st May to 30th April. Applications for membership should be made on the form at the end of the journal. Passes are required for entry to some Association events and for voting at Annual General Meetings. Applications for student membership will be accepted on a recommendation from a course supervisor. Overseas membership rates cover VECTOR surface postage and must be paid in UK £.

Corporate membership is offered to organisations where APL is in professional use. Corporate members receive multiple copies of VECTOR and are offered group attendance of Association meetings. Partaking individuals need not be identified but a contact person should be nominated 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 the editorial section of the journal and opportunities to inform APL users of their products via seminars and articles.

ADVERTISING

Advertisements in VECTOR should be submitted in typeset camera-ready A5 portrait format with a 20 mm blank border. Illustrations should be black-and-white photographs or line drawings. Rates are £200 per page. A6 and A7 sizes are offered pro-rata subject to layout constraints. Deadlines are:

Advertisement booking:	Beginning of June, September, December & March
Camera-ready copy:	1 week later. Distribution: Following month later.

Advertisements should be booked with and sent to Steve Lyus, whose address is given beneath the Index of Advertisers.

CONTENTS

Application Prototype Environment

an IBM program product for prototyping and developing screen-based applications under VS APL or APL2

This product is intended to be used both by the professional programmer and by business professionals, such as managers, planners, scientists, and engineers. The product is particularly suited for developing applications in Information Centre and Application Development Centre environments.

Three interactive functions are provided to easily and quickly define the end user interface.

Panel Design: For creating menu selection, action, and tutorial panels.

Chart Design: Charts (Plot, Surface, Bar, and Pie) are designed via an interactive dialogue.

File Handling: The interactive dialogue allows users to specify APL File, CMS, VSAM, QSAM, BDAM, and INSTORAGE access methods.

Over 70 cover functions are provided for building the application program. Some examples are:

Panel Operations

- Automatic Screen I/O.
- Automatic display of tutorials.
- Program function key control.

File Handling

- Reading/writing records from/to a file.
- Positioning the record pointer.
- Relative record access.

Object Library

The object library is used for storing, retrieving, and status checking of objects such as panels, functions, declarations, and variables.

Free 30-day Test Period

You have 30 days to test Application Prototype Environment to see how well it performs for you. If you are not completely satisfied, just send all materials back to your local IBM Branch Office within 30 days and you will not be billed.

For More Information: Contact your IBM sales representative or the nearest IBM Branch Office.

Technical Data

Application Prototype Environment is an IBM Licensed Program Product, Program Number 5668-896.

The program runs under CMS and TSO together with the following IBM programs or their equivalents: APL2 or VS APL, GDDM (Graphical Data Display Manager). Some examples of terminals supported are: 3277, 3279, 3270 PC/G and GX.

IBM

IBM United Kingdom Limited

EDITORIAL: THE MEN WHO RUN VECTOR.

by David Preedy

As the working group manfully struggles to put together another issue of VECTOR, I sometimes wonder how we are envisaged by the ordinary member on the Clapham omnibus. Clearly anybody with the awesome title of editor must be a power-crazed would-be press baron, eager only to use his position of influence to propagate his own highly prejudiced view of the world. What then of the rest of the working group?

There is of course the fearsome technical department - people, if such is the word, who only talk from right to left in sentences of one line. No doubt they spend their evenings huddled round cauldrons boiling up evil brews of devious inner products and vile concoctions from APL2. (Rumour even has it that they may understand parts of the I.S.O. standards.) And when they have nothing more rewarding to do, they pass the time dismembering innocent competition entries.

Looking round the rest of the working group, things hardly improve. There are people who have written articles, and even whole books, about APL. No doubt these authors lead a Bohemian existence in Bloomsbury, punctuated only by the occasion visit to the Royal Overseas League to throw a few pearls of wisdom before the assembled masses.

With such a daunting editorial group, a sane reader's instinctive reaction may well be to turn his attention to other more fruitful topics, such as how to incorporate the latest impossible user request into the system before the next monthly run this afternoon. "At least they don't want me to write an article for their journal. They wouldn't publish it and, after all, nobody wants to know about what I am doing. It's all very simple and doesn't even use an inner product. Anyway I don't spend much time programming." Perhaps similar thoughts have crossed your mind, so let me tell you about some of the people who really do want you to tell us all what you're doing.

First, there's the person who doesn't actually know much about APL himself. He happened to pick up a spare copy of VECTOR that was lying around the office and thought he'd take the chance to find out what all those funny Greek characters are about. What he'd like to read about is how APL is affecting the ordinary user, the range of application areas where it has proved its worth. Perhaps he might even take the plunge and try it for himself some day.

Then there's the sceptic from the computing department who has been brought up on the traditional D.P. approach involving projects measured in tens of man-years. He's heard claims about vast reductions in development times, developing closer links with the end-users, prototype systems, and so on, but wishes somebody would write about how this has been achieved in practice.

What about the APL expert? He's probably trying to build up APL expertise within his company, so that users can take increasing control over their projects and leave him free to tackle new and more challenging problems. He'd really like to know more about other people's experience in getting to grips with APL; what they find easy; what was hard; what sort of people naturally take to APL.

Finally, there's the working group ourselves. We want two things. We certainly want feedback about the sort of material you - the readers - are looking for in VECTOR. But equally we want to hear what you are doing with APL. Running the Journal is already a major occupation, and we certainly don't want to have to write all the text as well.

Of course, not everybody has the opportunity or the inclination to write a full-length article on what they are doing, although there are many whose "little project" would actually be of great interest to a wider audience. However, every reader must have views - favourable or unfavourable - about what they would like to see in VECTOR, and more generally the role they see for APL in the future. VECTOR is your Journal and we can only hope to keep it that way if YOU write in and tell us how you want it to develop.

Dates for future issues of VECTOR

	Vol. 2 No. 2	*Vol. 2 No. 3*	*Vol. 2 No. 4*
Copy date	9/8/85	25/10/85	31/1/86
Ad. booking	13/9/85	29/11/85	7/3/86
Ad. copy	20/9/85	6/12/85	14/3/86
Distribution	October	January	April

GENERAL CORRESPONDENCE

This section of VECTOR is reserved for communications of a general nature. Letters which require a high technical knowledge of APL or contain APL code usually appear in the Technical Section. Please note the requirements listed on the inside front cover. The editor reserves the right to edit letters unless the author states that a letter is to be published in full or not at all.

From Anthony Camacho 23rd March 1985

Sir: Many of us promote the benefits of APL among committed Cobollers, Pascallers and Fortanners and on the whole have very little success. A valuable weapon in such campaigning would be a graph plotting the cost of a job against the number of runs and showing an APL curve and curves for other languages on the same set of axes. The cost of course would be the sum of the program production and run charges.

I have been trying to collect some back of the envelope figures from my own experience and that of my acquaintances. It is surprising how very rarely any comparative estimating is ever thoroughly done. Most jobs are allocated to resources according to what is available or to who wants (or can be persuaded) to do the job. I would be glad to write an article for VECTOR about such comparisons, and would be grateful for descriptions and estimates on any job that has been estimated for APL and any other language.

If such a graph could be drawn (it will depend on whether the points plotted fall nearly enough on a line), it would be a very valuable weapon in the hands of those trying to get APL's virtues recognised. Please send comparative figures to me at 2 Blenheim Road, St. Albans, Herts., AL1 4NR.

Yours faithfully,

Anthony Camacho,
2 Blenheim Road,
St. Albans,
Herts., AL1 4NR.

(Editor: We would also like to widen the debate to include those who may not be convinced of APL's merits. If your evidence is more qualitative than quantitative, why not submit it for publication in VECTOR.)

From Dr. Heinz Reutersberg 13th June 1985

Sir: Please find enclosed my solution to the latest VECTOR competition. By the way, I consider VECTOR now the most interesting APL periodical, so keep up the good work!

Yours sincerely,

Heinz Reutersberg,
Cologne Reinsurance Company,
P.O. Box 10 80 16,
D-5000 Cologne.

From Savas Pavlides 12th May 1985
Sir: For years now I'm interested in computers and as for languages APL is my favourite. I do not yet have my own computer — they are a bit expensive, especially those which run APL. But now I'm student of Mathematics at the University of Patras, we have a big mainframe — a UNIVAC 1100/60 — which runs APL and I have the opportunity to learn and program with APL.

But I have found problems learning the language due to the shortage of books at the University library and the poor manual for UNIVAC's manual. I now have a book — Applied APL Programming by Wilbur LePage — but I want to have also other sources of information about APL. So when I read about VECTOR in another British magazine, I decided to write in order to get more information.

I'm the only APL user in our University, but now I have managed to get some students interested in APL, I don't want them to have the same problems that I had. Your magazine would be a precious help.

Yours sincerely,

Savas Pavlides,
17 Eleptherias str.,
Ampelokipi,
56123 Thessaloniki,
Greece.

(Editor: Keep up the good work in spreading the gospel at the University of Patras; at least you shouldn't have complaints about the funny Greek symbols! Perhaps the rest of the Greek APL community will get in touch with you. You may find the APL booklist gives you some more ideas for useful books. I haven't found any other British magazines referring to VECTOR, so you must read them more thoroughly than me!)

From Curtis A. Jones 10th June 1985

Sir: You've published a couple of fine puzzles in the January and May isues: easy enough for me to try; challenging enough to keep me going for a while. Here's what I've gotten. I can't wait to see other solutions. Thanks to all who have contributed to putting out VECTOR.

Yours,

Curtis A. Jones,
210 South 12th Street, apt.1,
San Jose,
California,
CA 95112.

(Note to technical editors: re Dick Bowman's letter in Vector Vol.1 No.4, could you try to make the competitions less interesting or next year we'll have no candidates for the committee.)

BRITISH APL ASSOCIATION NEWS

Minutes of the Annual General Meeting held at the Royal Overseas League, London on 28th May 1985

1. The minutes of the 1984 AGM were taken as read.

2. Chairman's review.
   (This is reproduced in full later in the British APL Association News section.)

3. Regulations.
   The draft regulations published in VECTOR Vol.1, No.4 are the result of extensive redrafting and no amendments had been received for discussion. They were overwhelmingly accepted by the meeting, though one member was unhappy that our affiliation made it necessary for our Chairman to be a member of the BCS.

4. Officers' reports.
   In the unavoidable absence of the Treasurer, the Chairman showed an outline of the accounts which will be published in full in VECTOR. They showed that the profit from "APL Business Technology 83" had enabled us to spend on VECTOR this year more than our income. We therefore have to increase our subscriptions by half and may well run at a loss this year in spite of this. The proposed subscription rates (plus a rate of £120 for overseas corporate members) were agreed without objection. The auditors were reappointed. Other officers felt that they need not add to what had been printed in VECTOR.

5. Elections to the Committee.
   The secretary reported that he had only had one advance nomination, who had not insisted on having his particulars circulated. A postal vote would have ben impossible without candidates. The following were elected:

Chairman	Dick Bowman
Treasurer	Mel Chapman
Secretary	Anthony Camacho
Journal	David Preedy
Activities	Stan Wilkinson
Education	Dick Gray
Technical	Dave Ziemann
Publicity	Romilly Cocking

6. Dick Bowman proposed a vote of thanks to Phil Goacher which was passed by acclaimation.

Anthony Camacho 6 June 1985.

British APL Association Chairman's Report, May 1985

by Philip Goacher

As outgoing Chairman, after three years in office, I look back over the past few years of achievement which have culminated in our Association becoming progressively international and about to host the annual APL conference for the first time in the UK.

In many ways we have changed a great deal, though in other ways we have changed very little.

From the original few members of the 'UK APL User Group' founded in 1976, membership grew steadily for some years. This was a predominantly UK based membership for which regular technical meetings were held at Imperial College in London with an annual one-day seminar and exhibition to review the APL hardware and software available.

When I became Chairman in 1982 we were in the interesting period when much was talked about APL on microcomputers, but all we had were specialist machines such as the MCM and the CP/M based Vanguard and early MicroAPL products.

At APL82 in Heidelberg we first saw APL2. This aroused a great deal of interest at very crowded demonstrations. At the same conference Inner Product unveiled VIZ:APL — a British *first* giving a virtual megabyte on a CP/M machine.

Appearing later that year was another home grown product — Dyalog APL, a 'second generation' interpreter developed to run on UNIX machines, with generalised arrays, virtual files and a lot more.

In 1983 the UK User Group ran its first conference at Loughborough. A successful first attempt which immediately raised the question as to whether we should host one of the annual international APL conferences following previous European venues such as Paris, Copenhagen, Pisa, Nordwijkerhout, and Heidelberg.

Not only did our success at Loughborough make the Group aspire to staging APL86, it also provided a substantial sum of money with which to develop the Group. A task force examined the structure of the User Group, its meetings and its newsletter. Its proposals were to revise the image of the Group, upgrade the quarterly newsletter and review the technical meetings.

In the last year all three areas have been tackled to some degree. The name of the Group was changed to The British APL Association which has undoubtedly helped to improve our image at home and overseas; new regulations have been drafted in keeping with a growing Association and the journal VECTOR was started from scratch. We are very concious that overseas members and those unable to travel regularly to our meetings rely on our publication for contact with the Association. Having spent a number of years helping Val Lusmore to produce a mere 32 page newsletter I know only too well the problems associated with gathering copy, editing and producing the final result. On behalf of the Association I would like to thank Robert Bittlestone and his Journal Editorial Committee for the tremendous work they have done in getting VECTOR off the ground so successfully as a leading APL publication. I bring many messages of congratulations from our overseas members who were at APL85 with their plea for you to keep it up.

The technical meetings have continued through the year, held recently at the Royal Overseas League in Central London, where we can provide more congenial surroundings for visitors. During the past year two special events were held in London. One brought some of the best speakers and exhibits from Helsinki to a seminar entitled 'APL84 in Focus', at the Waldorf Hotel.The second special event ('What's New for 1985' at the Regent Crest Hotel) provided an opportunity for the APL vendors to describe and exhibit their latest products and services.

Having successfully bid for APL86 whilst attending APL84 in Helsinki, we were fortunate in being able to send a team of three to APL85 in Seattle to promote next year's conference at UMIST. We are indebted to British Airways official carriers for APL86, who helped with travel arrangements and assisted with our promotion in Seattle.

And so to the present. We have a growing number of overseas delegates, both individual and corporate, as a result of publishing VECTOR, in addition to our UK members. We are now an international organisation with a widely read journal, but we should be planning to do even better — there are many more APL users to recruit, and you can help us do it.

We also look to our members to support the APL86 conference by submitting papers, by helping the committees involved in running the Association and the conference, by supporting VECTOR with articles and letters, and by coming to the conference next year.

I would like to thank all those who have served on the 1984 management committee of the Association for the support they have given me throughout the year. As retiring chairman I wish my successor and his committee every success in the coming year.

BRITISH APL ASSOCIATION
1985 REPORT AND ACCOUNTS

INCOME AND EXPENDITURE FOR YEAR ENDED 30th APRIL 1985.

	£	£
Income:		
Subscriptions:		
Full	3383	
Corporate	752	
Sustaining	2000	
Library	60	6195
Special Events:		
APL '84 in Focus	880	
What's new for 1985	762	1642
Total Income		7837
Expenditure:		
Meetings		1193
Vector (less advertising income)		11830
Committee		
BAA	713	
APL '86	350	1063
Mailing		1583
Professional fees		60
Total Expenditure		15729
Excess of expenditure over income		7892

BALANCE SHEET AS AT 30th APRIL 1985.

Cash	8828
Debtors	3532
	12360
Less Creditors	9722
Net assets	2638

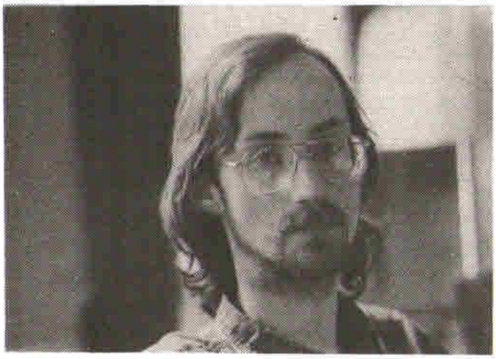
Note:

The committee will be making every effort to contain and, hopefully, reduce expenditure. It is hoped therefore that members will see the increased rates as more than justified and pay promptly.

The accounts, to BCS requirements, were prepared and signed by Mel Chapman as treasurer and audited by Roger Francis MSc. BSc.(Econ) ACIS.

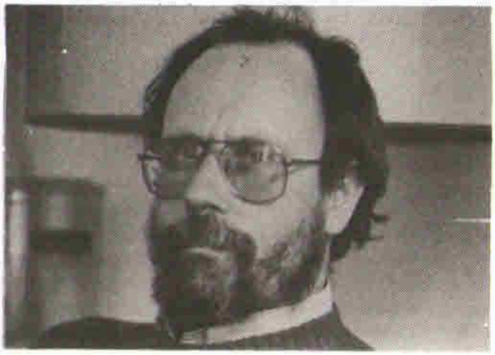
8th May, 1985

BRITISH APL ASSOCIATION COMMITTEE MEMBERS


Dave Ziemann
(*Technical*)

David Preedy
(*Journal*)

Romilly Cocking
(*Publicity*)


Dick Bowman
(*Chairman*)

Dick Gray
(Education)

Mel Chapman
(*Treasurer*)

Anthony Camacho
(Secretary)

Stan Wilkinson
(*Activities*)

Activities Programme 1985/6

Due to the careful advance planning of the Activities working group, the forthcoming Association activities remain as announced in VECTOR Vol.1 No.4:

September 13	— Large APL Systems There are increasing numbers of multi-person long term development projects within the APL community. What do we have to learn from the mainstream DP business and what in turn can we teach them?
October 18	— Micros and applications using micros.
November 15	— Compilers and Interpreters APL compilers are becoming a reality; what goes on inside them and what impact does this have on the application developer?

1986

January 17	— Joint Meeting with OR Society Who don't know about it yet! OR is an area which has had long links with using APL as a tool; what's going on and what can we both do to make OR use of APL even more successful and widespread?
March 21	— APL System Design Beyond wet fingers and pieces of string which design techniques are most useful to the APL developer; how much discipline is useful and how much is restrictive.
May 16	— AGM + Relational Databases etc. Besides your regular opportunity to vote with your feet, an examination of the impact which developments such as APL2's link with mainline DB2 databases may have on the future use of APL.
July 7 to 11	— APL86 Of which you will read much more elsewhere.

I would like to remind everybody about the questionnaire regarding the September meeting (should it be about ADRS?, if so should it be upgraded to a Special Event? or should it be Consultants Questiontime?); so far we haven't had many back.

Also on the theme of participation and making the Association match your needs, can I remind members of the sheer exultation into which the Activities Working Group is thrown whenever somebody spontaneously offers to stand up and talk at meetings. All you have to do is ask (allowing about two months' notice so that we can organize and publicise). Indeed we also welcome suggestions about subjects for possible meetings. So volunteer; after all the worst that's going to happen to you is that you'll be invited to join the Activities Group.

The venue for all Association meetings will continue to be the Royal Overseas League (London W1) with all meetings scheduled to begin at 2pm. Don't forget the opportunity which this gives you to meet other APL users and implementers both before and after the meeting proper.

Once again I remind everyone of the APL Publications bookstall which is a regular feature of London events and the opportunity which this gives you to purchase all that hard-to- find APL literature.

As usual, this programme is subject to revision; further details of each event will be sent to British members nearer to the date.

INTERNATIONAL APL NEWS

Compiled by David Preedy

This section of VECTOR is open to any APL society from outside the U.K. With the substantial, and growing, number of VECTOR subscribers from overseas, this column hopes to provide an effective way to keep APL users in touch with each other internationally.

We are very keen to hear of activities, conferences and publications sponsored by APL societies around the world. Unfortunately we seem to get most material too late for inclusion in the relevant issue of VECTOR. If any of our readers are involved in organising events, or simply attend as delegates, we would be pleased to publish reports and to give advance notice in the quick-reference diary.

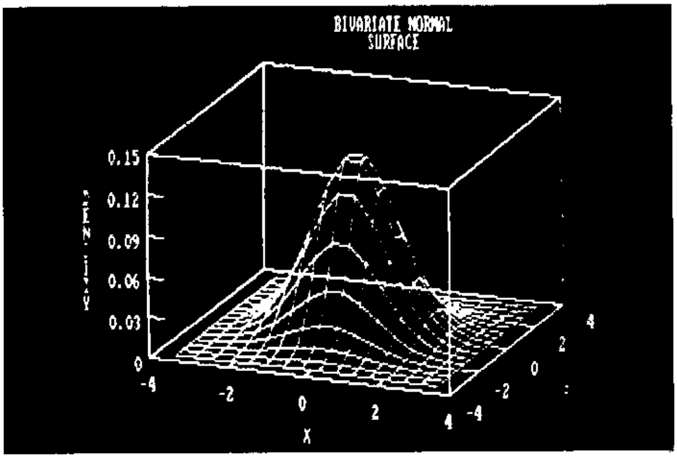
Swiss APL User Group

We have received with interest the latest copy of "APL-Notizen" — the bulletin of the Swiss APL User Group. Clearly the Swiss readership is expected to exhibit considerably greater linguistic skills than our own, as the newsletter switches effortlessly between German, French, English and of course APL. Apart from their editorial and announcements of meetings, the bulletin includes a technical article on the role of the "exclusive or" (with particular reference to list pointers) together with product news, book and article reviews, algorithms, quotations and cartoons. Anybody interested should contact the Group's president, Dr. Metzger, at the changed address:

Swiss APL User Group,
Dr. J.L. Metzger, President,
c/o Diametal A.G.,
Solothurnstrasse 136,
CH-2504 Biel,
Switzerland.
Telephone: (032) 41 39 71

STATGRAPHICS

Interactive Software for Data Analysis, Data Management and Statistical Graphics on IBM and Compatible P.C.s.



STATGRAPHICS is an integrated system of over 350 functions for interactive data analysis, data management and statistical graphics. It is designed to provide a flexible user-friendly environment for serious statistical computing, with system functions organised into 26 chapters covering all major areas of statistical analysis. Originally developed for use on large IBM mainframes, the system is now available for a variety of personal computers.

STATGRAPHICS is characterised by extensive integration of graphics directly into the statistical procedures. Graphical out-put may be generated by simply pressing selected function keys. Graphics displayed on the screen may be easily dumped to attached printers or routed to pen plotters. Selection of variables, transformations, and other options is accomplished by direct editing on fullscreen panels, rather than through sequences of complicated commands. The system has been designed to interface easily with popular micro software packages and to encourage the addition of user-written procedures. The modular construction of system functions, and easy access to native ASCII files creates an open-ended system which gives the analyst unlimited capabilities for exploratory analysis and manipulation of data.

MERCIA Software Limited Aston Science Park • Love Lane • Birmingham B7 4BJ
Phone 021-359 5096 Telex 334535

APL86 — 'APL IN ACTION'

7—11 July 1986

Manchester, England

Our plans for next year's conference are progressing well. We were able to get a team to APL85 in Seattle to promote the event with the assistance of British Airways — the official carriers for APL86. The response from the Seattle delegates was excellent and we look forward to welcoming many of them to England next year.

The theme of APL86 is ***APL in Action*** — emphasising the use of APL for a wide variety of applications with live presentations in support of the contributed papers wherever possible.

A comprehensive exhibition of APL hardware, software and literature will be on show throughout the conference. Product forum sessions will allow exhibitors to give public demonstrations of their products.

No international conference would be complete without a full social programme for delegates and accompanying persons. The North West area of England, around Manchester, has much to offer, including the lake district and peak district national parks and North Wales. In Manchester itself there is a wide variety of entertainment and good restaurants.

For overseas visitors, British Airways is developing a 'See England' package for those wishing to spend some vacation time before or after the conference. A fly-drive travel option can give you several days of touring around the country parks, stately homes, famous cities and picturesque villages that make England so attractive to visitors.

In forthcoming issues of VECTOR we will be publishing a draft conference programme based on the abstracts which are now arriving. We will also be giving more details of travel and vacation arrangements that you might want to take advantage of in order to make the most of APL86.

If you do not receive VECTOR regularly you can ensure that you are kept up-to-date by getting on to the APL86 mailing list. Also, if you are interested in being an exhibitor at the conference you should contact:

BISL Conference Department (APL86)
British Computer Society,
13 Mansfield Street,
London W1M 0BP,
England.

PAPERS FOR APL 86

The response so far to the call for papers has been excellent (over one hundred abstracts by mid-July). Not all of these will be suitable and of course we need more.

We asked for abstracts early because we have to plan the outline of the conference programme now. The plan has to be more or less fixed by the end of August. Things can be adjusted to accomodate papers whose abstracts are received early even if their topics are at the edges of our themes.

We will continue to accept abstracts and drafts and papers until there is no time left to get the refereeing done, but every week you delay increases the chances of your paper failing to get a place. We will be sending out details of layout and a style guide to authors in September and the current date for receipt of the papers is 1 November 1985. This is to allow our refereeing team time to deal with everything thoroughly and where appropriate to ask authors to amend their work to suit the neds of the conference. This activity must be completed over Christmas so that final acceptances can all be made for 10 January 1986. Anything, however good, you write after then will only find a place if someone else drops out.

If you hope to produce a paper, it will help your chances and our planning greatly if you let us know soon. Please send your name and address to me or to BISL Conference Department (APL 86) at the BCS and if you cannot devise an abstract let us know the subject or the field within which you intend to find a subject.

I believe there is something of interest to other APLers in most APL projects and that all you have to do is to find an effective way to present it. We are putting together a programme which will help all delegates to improve their own performance in APL and believe this is best done by sharing and discussing typical and ordinary applications on run-of-the-mill equipment as well as learning about work on next years machine or interpreter. Do not excuse yourself from sending in a paper on the grounds that your work offers nothing new.

We also want some controversy to add spice to the conference. Do you have a strongly held view which could be provocatively put and perhaps provocatively responded to? Please let us know soon if you have; we would like to include you in one of our debates.

We are organising better demonstration facilities than usual for APL 86 because live demonstrations are both more entertaining and more valuable to the APLer. It is easier to follow a demonstration on the screen than on a succession of slides or overhead projector foils as it shows more details of the sequence of events and should allow the demonstrator to show any detail of any function easily. The audience comes because it wants to learn about what you have done and how you did it and if it likes what it learns may want to adapt, emulate or improve on some or all of what is demonstrated. Do you have a good demonstration? We would like to hear of it.

The conference slogan is "APL in Action" and its other themes are "APL and its Environment" and "The APL Notation". We need to be able to advertise the conference as containing so much that will be valuable to APL users that employers will want to send everyone who can be spared. Keep 7-ll July 1986 free and persuade your colleagues to book their holidays at other times next year. Get them to send their papers in too. And help us to make it the best ever international APL conference. Send us your name and a promise a hope or a threat!

Anthony Camacho
Programme Chairman APL 86

NEWS FROM SUSTAINING MEMBERS

Compiled by Steve Lyus

"Sustaining Membership" of the British APL Association is available to any company trading in APL products or services. It provides a tangible way for such companies to express their commitment to APL and to promote increased interest and activity in APL and in the Association.

As well as receiving public acknowledgement for their sponsorship of the Association, sustaining members receive bulk copies of VECTOR for distribution amongst their customers and are offered the opportunity to submit news material for this section of the Journal. They are also able to submit announcements for Association meetings (subject to the approval of the Activities Officer) and are invited to inform APL users of their products via seminars and articles.

The Committee of the British APL Association would like to acknowledge the generous support of the following organisations that have become Association Sustaining Members.

APL People	17 Barton Street, Bath, Avon. Tel.0225-62601
APL Software Technology (UK) Ltd. Bristol, BS12 2PP.	14 Rosewood Avenue, Alveston, Bristol, BS12 2PP. Tel. 0454-415737
Cocking & Drury Ltd.	16 Berkeley Street, London, W1X 5AE. Tel. 01-493 6172
Dyadic Systems Ltd.	30 Camp Road, Farnborough, Hants., GU14 6EW. Tel. 0252-547222
Inner Product Ltd.	Eagle House, 73 Clapham Common South Side, London, SW4 9DG. Tel. 01-673 3354
MetaTechnics Systems Ltd.	Unit 216, 62 Tritton Road, London, SE21 8DE. Tel. 01-670 7959
MicroAPL Ltd.	Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London, SW8 5BP. Tel. 01-622 0395
I.P.Sharp Associates	10 Dean Farrar Street, London, SW1. Tel. 01-222 7033

Since last year we are pleased to welcome a new sustaining member, APL Software Technology, but have unfortunately lost two of last year's members. The APL community is not immune from the harsh realities of industrial life in the 80's.

COCKING & DRURY LTD.
THE APL PROFESSIONALS

Enhance your APL Information Centre

An enhanced APL will improve programmer productivity — but what if you can't afford to be cut off from future IBM Information Centre products?

Cocking & Drury have a range of powerful enhancements to VS APL that will not commit your company to a non-IBM strategy.

* Multi-user APL filing system
* Enhanced report formatting
* High speed printing
* Mailbox facilities
* Execution of non-APL programs from APL

For further information call us on 01-493 6172 or write to 16 Berkeley Street, London, W1X 5AE

APL Software Technology (UK) Limited

APL Software Technology has been busy extending its areas of activities beyond that of consultancy and support within the insurance and financial world into specific hardware and software products, utilising its extensive experience and knowledge gained across the mainframe, mini and micro scene.

They have just completed an assignment for a major computer manufacturer developing major portions of the MSDOS BIOS for one of their IBM type PCs, and this has greatly improved their understanding of the internals of micros and their operating system requirements and restrictions.

Their appointment as Ericsson dealers has allowed them to put together an 'APL Machine', the Ericsson PC enhanced by the PC Express board providing a speeded-up 8086 CPU and 512KB or 640KB RAM, together with the APL*PLUS/PC offering. This machine will process at speeds up to three times that of the equivalent 8088 CPU powered PC or XT.

In the software arena they can now offer the POWERTOOLS/PC product which has now completed site beta testing. This provides a compact 14K tool box of very fast and efficient assembler written functions that extend and enhance the APL*PLUS/PC product, allowing processing at typically 8 to 10 times the speed of the equivalent APL function.

The main feature of the product is its Forms Processor, which is not an AP124 Screen Manager replacement, but a powerful user / program interface around which to build user friendly window oriented applications, offering fast assembler written data handling facilities.

Cocking and Drury Limited

Cocking and Drury have recently taken on new staff. Beverley Satterley and Cindy Bavin join the company as office administrators. Allan Gay, previously with Centrefile, is currently providing support for Cocking and Drury's APL*PLUS mainframe customers. Shumit Rehman who comes from the Overseas Development Agency and Nick Telfer from I.P. Sharp Associates join as consultants. Ian Ford comes to the company to perform an industrial training year as part of his sandwich course at North Staffordshire Polytechnic.

The company successfully launched two software products at the recent PC User Show at Olympia. The long-awaited APL*PLUS PC system is now available on the Apricot, and the powerful statistical graphics package Stratgraphics was announced in the UK.

By the way, if you haven't got your Apricot poster yet — give them a call!

Dyadic Systems Limited

Dyadic has announced an implementation of Dyalog APLandDigital's new Microvax 2 running Ultrix-32. This system performs nearly as well as a Vax 780, and costs around £25,000. Dyalog APL is also being implemented on the MG.1 from Whitechapel Computer Works. This British system uses the NS32032 processor and has a built-in floating point unit and high resolution bit-mapped display. A 2Mb system with 22Mb disc and the dyalog APL interpreter costs just over £10,000.

Dyadic has appointed Pecoma BV as its exclusive distributor in Holland. Enquiries should be addressed to F. Smelik, Pecoma, Weesperzijde 84, 1091 EJ Amsterdam, Tel. 020-94 7561.

Dyalog APL has been chosen by BASF to run on its many HP9000 computers. Dyalog APL will be used in 'expert systems' for research chemists; an application for which its nested arrays and capability of interfacing to other languages make it particularly suitable.

MetaTechnics Systems Limited

On the new product front MetaTechnics Systems are soon to release MetaScreen II. Based on the successful MetaScreen (AP124 compatible full-screen management system for STSC'S APL*PLUS/PC), MetaScreen II contains a powerful new tool for the design and creation of formatted screens. Totally menu and command driven, the new module provides the ability to create a "panel" by painting it directly on to the screen. Complete editing facilities are available so that it is easy to modify existing screens.

MetaScreen II is now in Beta-test phase and product release is expected to be in July 1985. An upgrade kit will be available for existing MetaScreen users.

An agreement has been reached for Cocking & Drury to become dealers for MetaTechnics Systems' MetaProduct range. The range now includes MetaScreen, MetaPack (the comprehensive utilities package for APL on the PC) and MetaScreen II. Dealers for the MetaProducts are being sought in the USA, Europe and the UK.

MicroAPL

MicroAPL's news concentrates on two of their products — the Mirage/APL.68 000 software and the APL on the QL.

They have released the latest upgrades to Mirage/APL.68000, including APL.68000 Version 5.00, additional utility workspaces and Mirage utilities and APLKW — a completely new keyword version of APL.68000. The interpreter has been improved to incorporate significant speed-ups on some frequently-used operations — epsilon, dyadic iota and "and-dot-equals". The supervisor now allows a command line specification of APL, which adds some major new facilities to the APL environment. APL.68000 running under Mirage can now be invoked with a command line which allows a number of APL parameters to be set. This enables the user to specify printers, spoolers autoload workspaces, etc. The additional workspaces provide support for IBM 3270 protocol converters and check the integrity of the APL.68000 filespace.

In the first few months of its existence the symbolic version of APL for the QL has proved to be tremendously popular. It seems that real APL enthusiasts are keen to have access to the language at home for both personal and business use. Many companies too are finding 'real' APL on a cheap machine a welcome addition to their APL system.

I.P Sharp Associates Limited

I.P. Sharp are now settled into the new offices in Dean Farrar Street and this gives them much better facilities for demonstrations and courses as well as providing improved working conditions. The 6670 laser printer is operational and is fully integrated into their TEXTEDIT application package.

Sales of their Global Limits exposure management system for international banks are expanding satisfactorily and the product is now recognised as the world market leader.

Their international trading and market information systems of many kinds now provide a significant and growing proportion of their business. They expect the trend to be even more marked in the future due to the requirement for systems created by changes in the organisation of the London securities market.

The latest release of SHARP APL has been shipped to their inhouse sites providing significant improvements in resource usage, documentation and operating procedures as well as giving more facilities to application programmes.

APL Training Your Prayers Answered

We know what it's like. Your APL applications keep on growing. People get moved from department to department. More and more users want to do their own thing in APL.

You just can't keep up with the demand for APL training. Courses are too expensive. Your own time is too valuable. There simply has to be a better way to spread the word about APL.

Now there is. It's called **MicroSPAN/PC** and it's the world's best computer-aided training course in APL. Hundreds of people in the UK and abroad have learnt APL using it, and now it is available on the IBM-PC under APL*Plus/PC.

MicroSPAN/PC is interactive, so people can learn at their own pace. It's fun to use, with lots of diagrams and computer-aided jokes. It's very complete, and it's very thorough. Best of all, you just switch it on and it will guide anyone through APL – even if they've never used a computer before.

And it costs just £250 plus VAT. Less than the cost of sending one person on an introductory APL course. And much more effective.

MICRO
APL

MicroAPL Limited
Unit 1F, Nine Elms Industrial Estate,
87 Kirtling Street, London SW8 5BP.
Telephone: 01-622 0395

APL PRODUCT GUIDE

Compiled by Steve Lyus

VECTOR's exclusive APL 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.

This release of the Guide contains several additions of new products and suppliers. We do depend on the alacrity of suppliers to keep us informed about their products so that we can update the Guide for each issue of VECTOR. Any suppliers who are not included in the Guide should contact me to get their free entry — see address below.

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. Where no UK distributor has yet been appointed, the vendor should indicate whether this is imminent or whether approaches for representation by existing companies are welcomed.

For convenience to VECTOR readers, the product list has been divided into the following groups:

- Complete APL Systems (Hardware & Software)
- APL Timesharing Services
- APL Interpreters
- APL Visual Display Units
- APL character set printers
- APL-based packages
- APL Consultancy
- Other services
- Vendor addresses

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

Note: 'poa' indicates 'price on application'

All contributions to the APL Product Guide should be sent to:

Steve Lyus
Group Management Services,
Imperial Group plc.,
East Street
Bedminster
Bristol BS99 7JR

APL PRODUCT GUIDE

Company	Product	Prices £	Details
COMPLETE APL SYSTEMS			
A.S.T.	Ericsson PC	2000 – 5000	Authorised Ericsson dealer supplying complete APL systems based on Ericsson PC — fully IBM compatible including hardware add-on boards.
Cocking/Drury	MicroAPL SPECTRUM SAGE II SAGE IV	6000 – 35000	Supplied as part of a turnkey system. See MicroAPL entry.
Gen. Software	Myriade	poa	TI computer with APL and APL operating system
Inner Product	IBM PC	2000 – 6000	IBM PCs supplied for turnkey applications
MBS Rentals	IBM PC & PC/XT	2200 – 5000	For purchase or rental
	COMPAQ	poa	Portable IBM PC compatible
M.B.T.	MBT 10/40	poa	UNIX/68000 based multi-user APL system
	FORTUNE 32:16	poa	FORTUNE 68000 system enhanced for APL
	TORCH	poa	68000/Z80 multiprocessor
MetaTechnics	—	3000 +	Details on application
MicroAPL	Sinclair QL with QL/APL	from 434	Fully expandable APL system with colour graphics.
	SPECTRUM	11000 – 15000	Expandable multi-user APL computer using Motorola 68000. Standard configuration 1 Mb RAM, 12/36 Mb disc, 12 ports.
	SAGE IV	8500	Multi-user APL computer, 1 Mb RAM, 12/18 Mb disc.
APL TIMESHARING SERVICES			
Boeing	Mainstream APL	p.o.a.	Enhanced IBM VS APL (CMS)
Mercia	APL*PLUS	p.o.a.	STSC's Mainframe Service — MAILBOX etc.
I.P. Sharp	SHARP APL	p.o.a.	International Network application systems and public databases

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL INTERPRETERS			
A.S.T.	APL*PLUS/PC	600-695	See Cocking & Drury entry. Release 4 now available. Discounts available on multiple copies.
Burroughs	APL/700	3150	Runs on Burroughs B5000, B6000, B7000 and A9 mainframes.
Cocking/Drury	APL*PLUS/PC Release 4.1	695	Full feature interpreter for IBM PC, PC/XT, PC/AT & compatibles: CORONA, COMPAQ, COLUMBIA, WANG, OLIVETTI, EAGLE, AJ, ERICSON, ITT.
Dyadic	Dyalog APL	1000 – 8000	2nd generation portable APL for UNIX systems eg. VAX, HP9000, Fortune, MC68000, Zilog, Perkin-Elmer, Perq etc.
Gen. Software	APL*MYRIADE	poa	Runs on Texas Instruments TI990 range.
IBM(UK)	IBM PC APL	poa	With event-handling, & APs for full-screen I/O, disks, diskettes, asynch. comms.
Inner Product	VIZ::APL	250 – 350	8-bit Zilog Z-80 CP/M
	APL*PLUS/PC	600	See under Cocking & Drury
M.B.T.	Dyalog APL	poa	See Dyadic Systems entry
	MBTAPL	poa	Enhanced Dyalog APL for MBT hardware.
	VIZ::APL	poa	Customized for TORCH hardware
Mercia	APL*PLUS/PC Release 4.2	695	See under Cocking & Drury
	upgrade to 4.2	95	
	APL*PLUS/UNX	p.o.a.	Interpreter for UNIX Systems: WICAT, CADMUS, CALLAN, FORTUNE 32:16, HP 9000/500, OLIVETTI 3B2

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL INTERPRETERS (continued)			
MetaTechnics	APL*PLUS Release 4	695	Discount on quantity.
MicroAPL	APL.68000	1000+	Full implementation with component files, error trapping etc. for SPECTRUM, SAGE & other MC68000-based computers.
MicroAPL	QL/APL(keyword)	87	Full keyword APL for QL with many extra features.
	QL/APL(APL chars)	113	VSAPL compatible APL for QL with many extra features.
I.P. Sharp	Sharp APL	25000	For IBM mainframes
	Sharp APL/PC	55	For IBM PC or PC/XT
APL VISUAL DISPLAY UNITS			
Farnell	Tandberg TDV2221	1498	Ergonomic design APL terminal, 50-19200 baud, 15" anti-reflex screen, low profile keyboard.
	Tandberg TDV2271	1685	Combined APL/ANSI ergonomic terminal as above.
Gen. Software	Mellordata Elite 3045A	400	Second-hand.
Lynwood	Alpha	1995	Full APL character set, 16-bit microprocessor with pixel-addressable monochrome graphics.
	Alpha Colour	2985	As above with 8-colour graphics.
MBS Rentals	Lynwood Alpha	1995	See Lynwood entry.
	Lynwood Alpha Col.	2985	See Lynwood entry.
	Concept/APL	1491 – 1663	See Shandell entry.
	Concept GVT/APL+	2220 – 2393	See Shandell entry.
	(All above for purchase or rental)		

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL VISUAL DISPLAY UNITS (continued)			
M.B.T.	various		Contact MBT for details
MicroAPL	Insight VDT-1	795	Inexpensive APL VDU
	Insight GDT-1	1450	With monochrome graphics
	Tektronix 4105	4595	High resolution colour graphics supporting APL character set on MicroAPL hardware.
Shandell	AVT-APL+ GVT-APL+	1200+	ANSI 3.64; DEC compatible; full overstrike chars; 4/8 pages; 2/3 comms. ports; 80/132 cols; windowing; 12" screen; 46 PF keys; Tek 4013 graphics compatible.
	HDS 200	1200+	As above plus 15" screen; viewports; smooth scroll; NVM storage; 55 PF keys; Tek 4014, VT640/DQ640 & Visual 500 graphics compatible.
Sigmex	Sigmex 5684	from 6091	High-res colour graphics (768x512x4)
	Sigmex 5688	from 7096	High-res colour graphics (768x512x8)
	Sigmex 5472	from 6649	High-res monochrome (1536x1024x2)
	Sigmex 5484	from 10500	High-res colour (1536x1024x4) (Hard-copy output devices available for all these models.)
APL PRINTERS			
Datatrade	Datasouth DS180+	1295	180 cps matrix printer with 4K buffer, 9x7 dot matrix and APL option.
	Datasouth DS220	1695	Letter quality; graphics capability, APL option (both available with IBM Twinex or Coax interface).

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL PRINTERS (continued)			
Inner Product	Epson FX80	500	Soft char. set, 160 cps, 80 column
	Anadex 9620	1150	200 cps., 132 col., tractor feed
	Siemens PT88	620	180 cps., 80 col., silent
	TGC Starwriter	1180	40 cps., letter quality
MBS Rentals	Epson FX series	438-1100	
	Datasouth DS180	1395	See Datatrade entry.
	Phillips GP300	2251	Matrix with letter & draft quality & APL.
	TI745 + APL	1390	Portable dot-matrix.
	DEC LA120	2516	120 cps dot-matrix.
	DEC LA12C	1444	Portable dot-matrix.
	AJ833	2595	Daisy-wheel
	(All above for purchase or rental)		
M.B.T.	Facit 4565	poa	40 cps letter-quality
	Facit 4510/11/12	poa	Matrix printers
MetaTechnics	Quen-data	295	Low-cost APL daisy-wheel printer
MicroAPL	Datasouth DS180	1545	See Datatrade entry
	Phillips GP300	1798	Matrix printer with letter and draft quality and APL.
APL PACKAGES			
A.S.T.	POWER TOOLS/PC	275	Assembler written replacement function for commonly used CPU-consuming APL functions. Includes extended full-screen Forms Processor.
Boeing	TABAPL	p.o.a.	Hierarchical Planning System
Cocking/Drury	ARMS 2	3000 – 15000	Applications Generator
	AFM	10000	High performance shared file system.
	CALL/AP	3000	Non-APL program execution.

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL PACKAGES (continued)			
Cocking/Drury	FORMAT	2250	Enhanced report formatting
	WSAIDS	795	Workspace documentation and development aids.
E&S	PROTOPAK consisting of:		Packages for prototyping management information systems — PC & mainframe
	RMS	Modules	Relational databases.
	AMS	from 250	Multi-dimensional arrays.
	RAMS		Combined RMS & AMS.
	BMS		Dynamic financial modelling & forecasting
	FMS		Full-screen handler for IBM PC. (AP124-based)
	CMS		Communications package.
Gen. Software	PROPS	from 500	Spreadsheet system for Product and/or Project Planning.
H.M.W.	INPUT	poa	Matrix manipulation package for data entry & report generation
	PRINTPAK	poa	Block printing for VM/CMS
	VIEWPAK	poa	AP124 Protocol emulator for IBM/PC
Holtech	CASH	3500 – 10000	Accounting package and hotel management system on MicroAPL SPECTRUM & SAGE CPUs.
Inner Product	Viewcom	150	Control Viewdata from APL
	APL/dBASE II	150	Interface APL with dBase II
	APL/dBASE III	150	Interface APL with dBASE III
	APL/LOTUS	150	Interface APL with Lotus
	APL/WORDSTAR	150	Interface APL with Wordstar
	APL/MULTIPLAN	150	Interface APL with spreadsheet
	CEMAS	3500	EEC monetary and agrimonetary analysis.
M.B.T.	RHOMBUS	poa	Integrated Office System
	HASLEMERE	poa	Hotel Accounting System

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL PACKAGES (continued)			
Mercia	STATGRAPHICS	995	Integrated statistical graphic system for PCs.
	APL*PLUS tools VOL 1	295	IBM PC Utilities: IRMA 3270 comms, full screen, RAM disk report generator
	APL*PLUS tools VOL 2	95	File documentation, screen editing, Exception handling.
	FINANCIAL AND STATISTICAL LIB.	275	Financial and statistical analysis.
	INFO CENTRE	12,000—20,000	Full-screen entry, display and multi-dimensional analysis. Interfaces to other I.C. products. Runs under VM VSAPL on IBM mainframes.
	EXECUCALC	4,000	Mainframe Spreadsheet with VisiCalc and Lotus 1-2-3 functionality requires VSAPL under TSO or VM.
	EXECUPLOT	3,200	Mainframe Graphics display system with VisiPlot functionality requires VSAPL under TSO or VM and GDDM.
MetaTechnics	MetaScreen	245	Full-screen handler for APL*PLUS/PC, based on VSAPL AP124
	MetaPack	495	Comprehensive utilities package for APL*PLUS/PC. Includes MetaScreen, MetaWS, Browse, Toolbox, Numeric Editor
	ADAPTA DLS	poa	Production & purchasing scheduling for process manufacturing.
	ADAPTA MSP	poa	Job-shop loading & scheduling for multi-stage production.
MicroAPL	MicroTASK	250	Project development aids
	MicroFILE	250	File utilities and database
	MicroPLOT	250	Graphics for HP plotters etc.
	MicroLINK	250	General device communications

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL PACKAGES (continued)			
MicroAPL	MicroEDIT	250	Full screen APL editor
	MicroFORM	250	Full screen forms design
	MicroSPAN	250	Comprehensive APL tutor
	MicroGRID	poa	Ethernet & other networking
	APLCALC	400	APL spreadsheet system
	MicroPLOT/PC	250	For APL*PLUS/PC product
	MicroSPAN/PC	250	For APL*PLUS/PC product
Packer	Budworth Genealogy System	30	Record-keeping, documenting, sorting & pedigree-chart printing for professional & amateur family historians. Uses IBM/PC APL.
Parallax	ExecuCalc	$5000	Mainframe-based electronic spreadsheet for VM/CMS & MVS/TSO with links to micro products.
	ExecuPlot	$5000	Mainframe-based colour graphics system with micro links.
I.P. Sharp	ACT	poa	Actuarial system
	APS	poa	Financial modelling
	BOXJENKINS	poa	Forecasting technique
	CONSOL	poa	Financial Consolidation
	COURSE	poa	APL Instruction
	EASY	poa	Econometric Modelling
	FASTNET	poa	Project Management
	GLOBAL LIMITS	poa	Exposure management for banks
	MABRA	poa	Record maintenance/ reporting
	MAGIC	poa	Time series analysis/ reporting
	MAGICSTORE	poa	N-dimensional database system
	MAILBOX	poa	Electronic Mail
	MICROCOM	poa	Mainframe to micro link
	SAGA	poa	General graphics, most devices
	SIFT	poa	Forecasting system
	SNAP	poa	Project management
	SUPERPLOT	poa	Business graphics
	VIEWPOINT	poa	4GL — Info centre product
	XTABS	poa	Survey Analysis

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL CONSULTANCY (prices quoted are per day unless otherwise marked)			
APL People	Consultancy	poa	All levels, most APL systems
A.S.T.	Consultancy	poa	Technical & business systems, micros, networking & communications a speciality.
Attikale	Consultancy	160	Management reporting systems & Information Centres.
Boeing	Consultancy	poa	
Camacho	Consultancy	poa	Specialising in programming & manual writing.
Cocking/Drury	Consultancy	120-150	Junior consultant
		140-200	Consultant
		185-300	Senior consultant
		275-400	Managing consultant
	Courses	375	3 day Fundamentals
		330	3 day Advanced
		595	5 day System Design
Delphi	Consultancy	poa	Specialising in management reporting systems and APL on microcomputers.
Dyadic	Consultancy	poa	APL system design, consultancy, programming and training for Dyalog APL, VSAPL, APL*PLUS, IPSA APL etc.
E & S	Consultancy	150-250	System prototyping: all types of information system.
Gen. Software	Consultancy	from 100	
H.M.W.	Consultancy	100-250	System design consultancy, programming.
Inner Product	Consultancy & Training	200	On-site micro/mainframe APL, PC/DOS & Assembler

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
APL CONSULTANCY (continued)			
M.B.T.	Consultancy & Courses	poa	
Mercia	Consultancy & Training	poa	APL*PLUS & VSAPL consultancy. Onsite and Offsite APL courses.
MetaTechnics	Consultancy	poa	Management Information & Production Management Systems & Process Control.
	MetaSys	poa	Prototyping APL-Hybrid Custom Programming Service.
MicroAPL	Consultancy	poa	Technical & applications consultancy.
Packer	Consultancy	150	Specialising in Microcomputer & Operational Research applications.
Parallax	Consultancy & Training	$750 poa	Introductory APL, APL for end-user & Advanced Topics in APL.
QB On-Line	Consultancy	200	Specialising in Banking, Financial & Planning Systems.
I.P. Sharp	Consultancy	poa	Consultancy & support service world-wide.
OTHER PRODUCTS			
APL People	Employment Agency	poa	Permanent employees placed at all levels. Contractors supplied for short or long term projects, supervised.
APL Publications	Books	refer to APL Booklist	APL books.

APL PRODUCT GUIDE (continued)

Company	Product	Prices £	Details
OTHER PRODUCTS (continued)			
I.P. Sharp	Productivity Tools	poa	Utilities for systems, operations, administration & analysts; auxiliary processors, comms. software, international network.
	Databases	poa	Financial, aviation, energy and socioeconomic.

APL PRODUCT GUIDE (continued)

VENDOR ADDRESSES

Company	Contact	Address & Telephone
APL People	Valerie Lusmore	17 Barton Street, Bath, Avon. Tel. 0225-62601
APL Publications	Les Hollingbery	9 Koh-I-Noor Avenue, Bushey, Herts., WD2 3EJ. 01-950 1044
APL Software Technology (UK) Ltd. (A.S.T.)	John Hagger	14 Rosewood Avenue, Alveston, Bristol BS12 2PP 0454-415737
or	Per Hultin	46 Vicarage Hill, South Benfleet, Essex SS7 1PB. 03745-50501
Attikale Ltd.	Peter Hurdley	51 Brookside, Paulton, Bristol BS18 5YR. 0761-415427
Boeing Computer Services (Europe) Ltd.	Anne Harding	19 Fitzroy Street, London W1P 5AB. 01-631 0808
Burroughs Machines Ltd.	M.J. Fennel	Heathrow House, Bath Road, Hounslow, Middlesex TW5 9QL 01-750 1400
Anthony Camacho		2 Blenheim Road, St. Albans, Herts., AL1 4NR. St. Albans 60130
Cocking & Drury Ltd.	Romilly Cocking	16 Berkeley Street, London W1X 5AE. 01-493 6172
Datatrade Ltd.	Tony Checksfield	38 Billing Road, Northampton, NN1 5DQ. 0604-22289
Delphi Consultation Ltd.	David Crossley	Church Green House, Stanford-in-the-Vale, Oxon SN7 8LQ. 03677-384
Dyadic Systems Ltd.	Peter Donnelly	30 Camp Road, Farnborough, Hants. GU14 6EW. 0252-547222
E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS. 0689-24741

APL PRODUCT GUIDE (continued)

VENDOR ADDRESSES

Company	Contact	Address & Telephone
Farnell International Instruments Ltd.	R. Fairbairn	Jubilee House, Sandbeck Way, Wetherby, W. Yorks. 0937-61961
or	Roger Attard	Davenport House, Bowers Way, Harpenden, Herts. 05827-69071
General Software Ltd.	M.E. Martin	22 Russell Road, Northolt, Middlesex UB5 4QS. 01-864 9537
H.M.W. Programming Consultants Ltd.	Ken Jackson	142 Feltham Hill Road, Ashford, Middlesex TW15 1HN. 07842-41232
Holtech Ltd.	Jan Bateman	'O' Block 4th Floor, Metropolitan Wharf, Wapping Wall, London E1 9SS. 01-481 3207
IBM UK Ltd	Gerard Johnstone	PO Box 32, Alencon Link, Basingstoke, Hants., RG21 1EJ. 0256-56144
Inner Product Ltd.	Dominic Murphy	Eagle House, 73 Clapham Common Southside, London SW4 9DG. 01-673 3354
Lynwood Scientific Developments Ltd.	Gareth Wokes	Park House, The High Street, Alton, Hants GU34 1EN. 0420-87024
M.B.S. Rentals	Robert Notley	119/120 High Street, Eton, Windsor, Berkshire. 07535-68171
Mercia Software Ltd.	Gareth Brentnall Barrie Webster	Aston Science Park, Love Lane, Birmingham B7 4BJ. 021-359 5096
MetaTechnics Systems Ltd.	Steve Margolis	Unit 216, 62 Tritton Road, London, SE21 8DE. 01-670 7959
MicroAPL Ltd.	Richard Nabavi Paul Thornton	Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London SW8 5BP. 01-622 0395

APL PRODUCT GUIDE (continued)

VENDOR ADDRESSES

Company	Contact	Address & Telephone
Modern Business Technology Ltd. (M.B.T.)	Michael Branson	P.O. Box 87, Guildford, Surrey GU4 8BB. 04868-23956
S.H. Packer		Cleveland House, Little Budworth, Tarporley, Cheshire, CW6 9BP. 082-921 439
Parallax Systems Inc.	Kevin Weaver	60 West 9th Street, New York, New York 10011, U.S.A. 212-475-4001
QB On-Line Systems	Philip Bulmer	5 Surrey House, Portsmouth Road, Camberley, Surrey, GU15 1LB 0276-20789
Shandell Systems Ltd.	Maurice Shanahan	12 High Street, Chalfont St. Giles, Bucks HP8 4QA. 02407-2027
I.P. Sharp Associates Ltd.	David Weatherby	10 Dean Farrar Street, London SW1 01-222 7033
Sigmex Ltd.	Alan Taylor	Sigma House, North Heath Lane, Horsham, W.Sussex, RH12 4UZ. 0403-50445

BOOKLIST

LIST OF BOOKS STOCKED AND ON SALE: NOTE SPECIAL PRICES FOR MEMBERS

Description	Invoice	Cash with order	British APL Assn Members Invoice	British APL Assn Members Cash	UK Post & pack
	£	£	£	£	£
APL: An Interactive Approach, Gilman & Rose	20.71	20.21	19.50	19.00	2.65
APL: An Introduction, H. Peele	11.20	10.80	10.40	10.00	2.25
Introducing APL to Teachers, K. Iverson	.80	.75	.70	.65	.40
Introduction to APL for Scientists & Engineers	.80	.75	.70	.65	.40
An APL Notebook, Barrie Wetherill	1.90	1.60	1.30	1.00	.50
A Source Book in APL, A Falkoff & K. Iverson	8.75	8.28	8.20	6.96	1.80
APL: Design Handbook for Commercial Systems, Smith	12.25	12.00	10.35	10.00	1.50
APL and Insight, P. Berry and G. Bartoli	4.50	4.20	3.60	3.45	.55
APL in Practice, STSC	20.00	18.00	16.00	15.00	2.50
APL The Language & its Usage, R. Polivka & S. Pakin	29.95	28.95	27.95	26.95	2.70
APL Quote-quad the Early Years	31.50	31.00	30.00	29.60	2.70
APL Business Technology '83 Proceedings	11.20	10.80	10.40	10.00	2.70
APL SV Reference Card	.30	.28	.25	.23	SAE
Designing APL Systems Toronto Procs 82 Vol 2	6.50	6.00	5.50	5.00	1.80
FinnAPL Idiom Library	11.20	10.80	10.40	10.00	1.30
Sharp APL Reference Book	1.30	1.20	1.10	1.00	.50
Sharp APL Reference Manual, P. Berry	10.00	9.70	9.30	9.00	2.70
Yale University Idiom list	1.50	1.40	1.30	1.00	.50
APL in Exposition, K. Iverson	.95	.90	.85	.80	.50
Algebra, K. Iverson	7.95	7.65	7.40	7.10	2.30
Solutions to Algebra, J. Iverson	2.00	1.80	1.70	1.50	.50
Calculus in a New Key, D. Orth	6.85	6.60	6.35	6.10	1.70
Elementary Analysis, K. Iverson	5.40	5.20	5.00	4.80	1.50
A Case Study in BASIC, APL and Functional Prog.	4.80	4.60	4.40	4.30	1.00
Resistive Circuit Theory, R. Spence	10.30	10.00	9.55	9.20	2.80
Star Map, P. Berry & J. Thorstetsen	3.50	3.20	3.10	2.90	.50
Reliable Software Through Composite Design, Myers	9.90	9.65	9.30	9.00	1.30
APL Lapel Pin with gripping backplate	2.00	1.90	1.70	1.60	SAE
Whizzbangs Volume II, R. Sykes	6.90	6.60	6.30	6.00	1.80
Whizzbangs Volume I, R. Sykes	6.90	6.60	6.30	6.00	.40
Information Centres Toronto Procs 84	8.00	7.50	7.00	6.00	1.80

PLEASE ORDER DIRECT FROM APL PUBLICATIONS

APL Publications: L.R. Hollingbery, 9 Koh-I-Noor Avenue, Bushey, Herts, WD2 3EJ
Telephone: 01-950-1044 evenings 1800 to 2100.

This list replaces all previous lists.

Prices are in pounds sterling and are as as 1st July 1985
Terms: sterling cheque with order; invoices payable on receipt of books.
Non-UK postage is extra: please specify AIRMAIL or SURFACE MAIL.

Save postage: a selection of books is usually on sale at APL Association meetings.

PRODUCT REVIEWS

by David Preedy

Hewlett-Packard 7550 8-pen plotter.

One of areas where APL has made a major contribution to business computing has been in introducing quality graphics to improve the presentation of corporate information. Despite improvements in ink-jet printers, the pen plotter still provides the best quality colour hard-copy. So in view of Hewlett-Packard's long established reputation in the plotter market, it was with high expectations that I set down to use their latest offering.

The 7550 allows you to have up to eight different pens per carousel, giving a choice of colours and/or line thickness. It can handle A3 or A4 paper, and the latter can be automatically fed from a stack underneath the plotting-bed. The plotter uses the latest grit-wheel technology, which means that the pen moves in one direction only, and the paper is moved as required in the perpendicular direction. By simplifying the mechanism and reducing the number of moving parts, this should improve reliability substantially. It also allows pen velocities and acceleration to be significantly greater then previous plotters.

The HP7550 has different carousels for the standard fibre-tip pens, roller-ball pens, liquid-ink drafting pens and special pens for overhead transparencies. On its own, the use of carousels is a major advance, as it removes the need to be continually changing individual pens if you switch frequently between paper and acetate copies, and the carousels give you somewhere convenient to store the pens not currently in use. But H-P have built even more functionality into the carousels. They control the default settings of the pen velocity and pressure, so that the plotter will automatically slow down and reduce pressure when drawing on acetate. So the choice of what plotting medium to use can be made locally on the plotter without having any implications for the software generating the graphics.

In fact the whole area of software requirements seems to have been well thought out by Hewlett-Packard. The 7550 uses HPGL — Hewlett-Packard's own graphics language; this means that there is a reasonable chance that code written for other recent Hewlett-Packard plotters will work without needing changes, although you may find that you want to modify your software to take advantage of any new features.

If you do have to start from scratch, then HPGL makes the programming reasonably straightforward. No longer do you have to write complex encoding routines to compress the x-y coordinates of a point into the minimum number of bytes. HPGL takes coordinates represented as ASCII text. Similarly you can throw away that 30-line function you used to shade an arbitrarily shaped polygon. The 7550 firmware provides automatic polygon-shading, giving you control over the line-patterns, angles and separation used for shading. If you want multiple copies then you can simply send a command to repeat the last chart up to 99 times; the copies will be done locally without tying up your computer.

With a little bit of care in the design of your software, you can arrange for character sizes, patterns, etc., to be defined relative to the size of your plotting area. This enables you to change the plotting area on initialisation of a chart (e.g. to use A3 paper, or to fit several charts onto one page), and the chart will be automatically scaled and rotated without you having to change your program.

To my mind, however, the key requirement of a device like this is that it can run under user control with the minimum operator involvement. In the past this has required the use of roll-feed paper with an automatic guillotine. Not only was the paper-feed mechanism notoriously unreliable, but the finished graph ended up on special paper that was a different quality from the rest of your report and could never be pressed flat. The A4 paper stack-feed overcomes these problems. It seems to be very reliable if you load it carefully and try to make sure that the leading edge of the paper is not creased. But the great advantage is that you can load the same quality paper that is used for the rest of your report, for instance using your company's headed stationery.

So far our use of the HP7550 has highlighted only two significant complaints. The first arises from the use of grit-wheels to move the paper backwards and forwards. This has two side-effects. On normal paper you tend to get two slightly rubbed areas along the sides of the chart. More importantly, you cannot use conventional acetate for overhead transparencies because the grit-wheels will not grip; instead you have to buy special acetate which Hewlett-Packard has treated to have roughened areas where the grit-wheels can grip.

The second complaint concerns the loop-through arrangement, whereby you can put the plotter in the line between your computer and your terminal. Unlike previous Hewlett-Packard plotters, the 7550 disables this link when it is turned off. Consequently you either have to rewire your link or put up with the the fan-noise (and heat) generated by the plotter continuously. Moreover, even when it is turned on, the plotter accepts 8-bit characters from the computer, but only transmits 7 bits on to the terminal, and it does not transmit any hard-wire handshaking to or from the terminal.

In summary, the new HP7550 sets new standards by which the current generation of plotters can be judged. Hewlett-Packard seem to have learnt from the shortcomings of their previous plotters and this latest model exhibits many excellent design features. The trend to increase the local power of the plotter will presumably continue and this enables micro users in particular to achieve significant improvements in performance. My main regret is a nostalgic one; no longer can you see the pen writing out the characters using the left- handed style of the old HP7221!

Hewlett-Packard 2686A LaserJet printer.

For many years there has been a straightforward choice for low-volume printers. If you want reasonable speed, you choose a dot-matrix printer; for "letter-quality" output you need a daisywheel printer. Over the past year though a third option has emerged in the form of low-cost laser printers. Many manufacturers' names may appear on the front, but these represent different offerings of a couple of basic printing engines. The LaserJet is one of several models based on the Canon engine.

The basic LaserJet provides excellent quality printout, at a reasonable speed (8 pages per minute). A growing range of font-cartridges are available, and these let you mix several different fonts onto one page of output. Typically you may want to have normal, bold and italic text. The maximum paper size is A4, which can be fed automatically from a stack held at the bottom of the printer. This can be overridden by a manual feed at the back and this can take labels, envelopes, etc. There is a rudimentary graphics facility (monochrome only of course) which may be suitable for drawing boxes but would need some fairly hefty software to do anything much more interesting.

The main attractions of the LaserJet as an office machine are the superb quality of the printed output (a resolution of 300 dots per inch is claimed), the A4 stack feed (for the same reasons as given above for the HP7550 plotter) and the fact that it is virtually noiseless. It looks much more like a small photo-copier than a printer and its noise level is similar. You can also send a command to request up to 99 copies of the current page should your photo-copier be out of action.

The range of fonts is extensive, but it is worth looking closely at the exact specifications of the cartridges to make sure that the font combinations you want are available together. Some of the literature gives a misleading impression of how many combinations are available. Fonts are defined by several attributes — orientation, symbol set, spacing, point size, style, stroke weight and typeface. This gives some 7500 possible different fonts but a typical cartridge may contain 10 or so.

This flexibility does have some unfortunate side-effects. In order to select a font, you can't simply say "select font number 3 from the loaded cartridge". Instead you have to specify enough of the font attributes to uniquely define a font available on the cartridge you currently have loaded. Typically, this means that you may have to send a 25-byte string to the printer to ensure that it loads the correct font. This is simply a chore if you are working direct from APL, but is normally impossible using a package unless it has been designed with the LaserJet in mind. In particular, I doubt whether any word-processing packages have yet built in the capability to use the LaserJet's facilities effectively.

We have had to develop an APL front-end in order to get Wordstar to change fonts and even then we don't have the complete flexibility that we would like. We cannot easily combine more than 2 fonts per document and we cannot get justified text using the proportionally spaced fonts. Probably a good Wordstar interface will be developed sometime, but I suspect that it is likely to be later rather than sooner. We have heard of two companies working on the link, but we have yet to see something working.

Whilst I cannot envisage the LaserJet being used for bulk APL listings, its ability to mix fonts suggests a useful role for preparing high-quality mixed APL/ASCII output — either as camera-ready copy for all the budding Gilmen and Roses preparing their next tome, or for those all-important VECTOR competition entries. Whilst Hewlett-Packard do not to my knowledge offer an APL character set, a company called Electronic Printing Services do provide a range of different fonts, including APL.

(Ed — we would welcome contributions from anyone with experience of using this or similar APL fonts on laser printers.)

We have had a few grumbles about some aspects of the LaserJet. I think the most significant is the absence of any facility to download a font definition from your host computer. This capability would open up all sorts of options for specialised fonts; two suggestions that spring immediately to mind are APL symbols and a character set with the pound sign somewhere vaguely resembling its position on a U.K. keyboard. My other main complaint is the LaserJet's refusal to fit 80 columns of text onto an A4 sheet using 10 pitch character spacing. It fits 77 columns on and there would be room for the other three, but if you want to dump your VDU contents you have to select a non-standard font.

On the whole reliability has been pretty good, but the printer did have one spell when it seemed unable to orientate the lines parallel to the edge of the paper. Paper-feeding has been of intermittent quality with occasional days when the printer refuses to feed heavier grades of paper and sometimes using lighter grades of paper, it picks up two sheets at a time. It tends to be particularly unreliable when the paper feed is almost full or nearly empty. On the good side, it is straightforward to clear a misfeed (unlike the majority of photocopiers seem to be), and the hot parts are not only suitably labelled and insulated, but are also coloured a lurid, almost luminescent green.

We have attempted to create overhead transparencies through the machine (the manual says it works) and, to be fair, it did create a good image on the acetate. But the heat of the printer caused the acetate to buckle a lot, and the dread of having to extract a molten sheet of acetate from the inside of the printer has deterred me from experimenting much further with it.

Obviously the LaserJet is considerably more expensive than a daisywheel printer, but in return it offers much better quality, is much quieter and needs much less maintenance. Unfortunately, I think it will take some time for software vendors (of word-processing packages in particular) to come to terms with the options offered by this device, and I am sure that will be this lack of suitable packages that will limit the attractions of the LaserJet for many users.

RECENT MEETINGS

This section of VECTOR is intended to document recent meetings of the Association, particularly for those members who work outside London and often find it hard to spare the time to attend. We also include reports of other meetings with an APL theme.

We are dependent on speakers for their willngness to provide us with a written version of their talks and would remind them that "a picture's worth 1000 words". Copies of slides and transparencies will enhance their articles.

The Activities officer (details on inside back cover) will respond enthusiastically to offers from individuals to contribute seminars and supporting papers.

INTRODUCTORY NOTES

by Adrian Smith

This issue of VECTOR is mainly devoted to the April workshop on Nested Arrays. For an overtly technical meeting this was extremely well attended, and once again the Royal Over-Seas League came up trumps with excellent facilities. I am very grateful to David Crossley for providing a full write-up of his most interesting talk, and I remain hopeful that we may yet see a more structured version of Graeme Robertson's fascinating peregrination through the grammar of nested systems!

As for the AGM, our own very brief notes (for which many thanks to my colleague Eileen Dyson) could not attempt to cover the details of what is basically a highly technical subject, *viz* connecting mainframes and PCs. We are therefore pleased to be able to include Graham Fieldsend's paper on this valuable but often less-than-riveting topic; there seem to be so many ways of doing things, and none of them seem particularly satisfactory.

Nested Arrays Workshop April 26th 1985

The Royal Over-Seas League

by Adrian Smith

Introduction

A most invigorating meeting, with four contrasted but fascinating papers. David Crossley began with a useful summary of the logical structure of the nested systems, and then went on to show how it was physically implemented on APL*PLUS. My interest had already been aroused by some examples in the Interprocess Newsletter (the topic was the extension of their AFM file package to include APL2) which showed how a 7-element character vector could be combined with a 2 by 3 integer matrix to occupy an astonishing 95 bytes of storage! For more details, have a careful look at David's paper, which follows these notes.

APL2 and SQL/DS

David Bryant (IBM)

To some extent, this talk assumed a basic knowledge of what SQL is and does. My impression is that it provides a very high-level query capability (of the 'FETCH this WHERE that OR the other' variety) for good old fashioned batch languages like PL/1. It applies this language to IBM's relational database (called DB2) to give programmers an extremely powerful tool for application development. Many SQL expressions look so like the sort of things we write in APL that I seriously wonder if APL was used to prototype it!

David first made the point that the interface to SQL is pretty well the same under VM/CMS and TSO. It effectively gives 'pipeline' access to SQL commands, and a few simple cover functions will make APL access to SQL look just like PL/1 or whatever. However APL has one great advantage over other languages when it comes to dealing with the results of the SQL operations: typically what you get back are subsets of tables; poor old PL/1 then has to loop through these row by row; in APL you can generally process them as they stand.

This is in part because the SQL structure (any element can be a number, a character or null) is just a simple form of an APL2 array! This lets you do block retrievals directly into APL objects extremely cleanly and efficiently.

Naughty (non-IBM) thought: was the APL2 array philosophy (in particular the use of heterogeneous arrays) designed with SQL heavily in mind? The interface is so 'made to measure' that I can't help feeling that it was. This may be a good thing in the short-term, but surely it must compromise the future of APL. Does anyone else have any views on this? (ACDS)

The Grammar of Nested APL

Graeme Robertson (IPSA)

My notes on this talk are succinct and to the point: " ...eek - has GR finally flipped?!" is all I could manage to record about this extremely entertaining half hour. It was one of those talks which probably has some significant subliminal effect on your whole outlook on APL; the impact on the conscious and rational part of the brain was (at least in my case) close to zero. I hope Graeme will try to write it down, and hit us with a technical paper sometime soon. It should make good bedtime reading.

Displaying Nested Arrays

by John Scholes (Dyalog)

Audience participation was the key element in this 'talk'. John sat behind a keyboard, and responded to a battery of questions about how one could format simple reports using nested structures.

His basic message was that plain vanilla APL is easy enough when all you want is '2+2', but that it really gets in your way (and up your nose) when you want to format a simple report. You end up with so much catenating and reshaping (not to mention the need to 'format' all the numbers) that even the most trivial report can require quite an investment in time and brain cells.

With nested structures, at least the simple things are once again made simple. To get headings, stubs and footers around a table of numbers you just catenate them on the appropriate axes, and a simple monadic 'format' will do the rest. The way John built these up as he went along was most impressive, all the more in that he clearly wasn't following a preset script.

What was also interesting was that there is still a cutoff point beyond which you once again get into the realms of re-shaping, and lots of nasty looking 'takes' with shoals of brackets. I can't remember exactly where this occurred, but it seemed to me to lead straight from 'very easy' to 'very hard' as the nested structure suddenly started to get in the way, and you needed to make sure everything was at the right *depth* as well as being the right shape. Maybe John would like to come back at me on this.

THE IMPORTANCE OF BEING NESTED

David Crossley

Nested array versions of APL have been publicly available now for over three years. In this paper, I present some views and guidelines based on continuous experience over this period of using APL*PLUS's NARS (Nested Array Research System) and Dyalog APL.

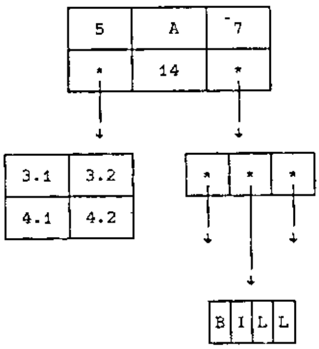
What Is A Nested Array?

I begin with the basics on the assumption that nested array theory is not a familiar subject. The diagrams below illustrate a traditional array and a nested array each in the shape of a matrix:

Traditional Array

5	4.2	¯7
10	14	0

Nested Array



Familiar attributes of the traditional array are that:

— it is rectangular.
— it is homogeneous, i.e. numeric in this case.
— each element is a simple scalar item.

The nested array is also rectangular. However, an element may be an array that is enclosed to form a NON-SIMPLE scalar. The array that is enclosed is called the ITEM. Thus a nested array is a rectangular structure whose elements may be simple or non-simple scalars and whose items may be simple scalars or arrays. If all the elements are simple scalars, it looks like, in fact is, a traditional array.

Unfortunately there are two fundamentally different theories defining nested arrays. Briefly, the difference centres around the treatment of simple scalars. The "grounded" theory (advocated by Iverson, Bernecky et al and implemented in Sharp APL) insists that enclosing any array results in an extra level of nesting. The "floating" theory (supported by More, Brown, Smith, Scholes et al and implemented in APL2, APL*PLUS and Dyalog APL) considers a simple array as a special case, and the result of enclosing is the same simple scalar.

Sharp APL does not allow a nested array to include simple scalar elements; all elements must be nested to a DEPTH of at least one (a simple scalar has a depth of zero). However, versions based on the "floating" theory allow simple scalars to float to the topmost level, a definition that permits simple arrays with both text and numeric elements — called a MIXED or HETEROGENEOUS array.

Enclosing Arrays

Enclose is a new function that produces a scalar when applied to an array. Some results of applying this and other new functions are illustrated below for the different implementations — observe also that different symbols are used. Since display methods vary, I have used boxing to indicate levels of nesting.

Action	"Floating" theory APL2/APL*PLUS/Dyalog	"Grounded theory Sharp APL
Enclose of a simple scalar	⊂10 10	<10 $\boxed{10}$
Enclose of an array	⊂1 2 3 $\boxed{1\ 2\ 3}$	<1 2 3 $\boxed{1\ 2\ 3}$
Catenation of nested arrays	(⊂1 2),⊂3 4 $\boxed{1\ 2}\ \boxed{3\ 4}$	<(1 2),<3 4 $\boxed{1\ 2}\ \boxed{3\ 4}$
Catenation of simple scalar and nested array	1,⊂3 4 1 $\boxed{3\ 4}$	1,<3 4 DOMAIN ERROR

	S←⊂10
	V←(⊂1 2 3),⊂4 5 6
		S←<10
		V←(<1 2 3),<4 5 6

Disclose - also called First	⊃S 10 ⊃V 1 2 3	>S 10 >V 1 2 3 4 5 6
Split - similar to > in Sharp APL	↓V 1 2 3 4 5 6	not implemented - see Disclose

Strand Notation

Strand notation is the term used to define a vector formed from two or more adjacent items. We already have this capability with simple items, eg

```
IV ← 10 20 30

TV ← 'ABC'
```

The extension of this principle allows:

```
NV ← e1 e2 e3.... eN
```

where el is any APL expression whose result is an array. Parentheses may be necessary to separate expressions. Strand notation appears to be incompatible with the "grounded" theory. It is implemented in all "floating" theory versions. Formally, it is equivalent to:

```
NV ← (⊂e1) , (⊂e2) , (⊂e3) , ..... ,(⊂eN)
```

Which would you prefer to write? Some examples are:

```
      IV TV

┌────────┐ ┌───┐
│10 20 30│ │ABC│
└────────┘ └───┘


      (2+2)(4×12)

4 48                          - note the scalar results


      'FRED' 'BILL' 'JIM'

┌────┐ ┌────┐ ┌───┐           - APL*PLUS requires parentheses here
│FRED│ │BILL│ │JIM│             but this is soon to be relaxed
└────┘ └────┘ └───┘
```

Strand notation also applies to assignment:

```
                  A B C ← ⎕ ← 'ONE' 2 (⊂,1+2)

              ┌───┐     ┌─────┐
              │ONE│  2  │┌─┐  │
              └───┘     ││3│  │
                        │└─┘  │
                        └─────┘

                  A            B             C

              ┌───┐                        ┌─────┐
              │ONE│           2            │┌─┐  │
              └───┘                        ││3│  │
                                           │└─┘  │
                                           └─────┘
```

The following example illustrates strand notation usage to form a nested vector of inverted items:

```
NAME (TM)              AGE (IV)             SKILLS (IM)
---------              --------             -----------
W.SMITH                  22                 2 5 7 0 0
P.JONES                  45                 1 2 6 7 9
J.LEWIS                  31                 5 7 8 0 0


      PERSONNEL ← NAME AGE SKILLS

      0 1 0 ⌿¨ PERSONNEL

┌───────┐  ┌──┐  ┌─────────┐
│P.JONES│  │45│  │1 2 6 7 9│
└───────┘  └──┘  └─────────┘

      PERSONNEL ← PERSONNEL JOIN1 ¨ ('D.BERRY') 19 (1 3)

      PERSONNEL

┌───────┐  ┌──┐  ┌─────────┐
│W.SMITH│  │22│  │2 5 7 0 0│
│P.JONES│  │45│  │1 2 6 7 9│
│J.LEWIS│  │31│  │5 7 8 0 0│
│D.BERRY│  │19│  │1 3 0 0 0│
└───────┘  └──┘  └─────────┘
```

Various criticisms are made against strand notation:

a) that there is an implied function between each strand element.
b) that visual context is further confused (what does A B C F X Y Z mean?)
c) that a single-element vector cannot be formed without explicit use of two functions, ie ravel enclose.

Note that for point (c) the same applies for a simple input; assignment of a scalar results in a scalar, not a vector. However, a point to watch is the case of a single expression when using execute to build a strand, as eg.

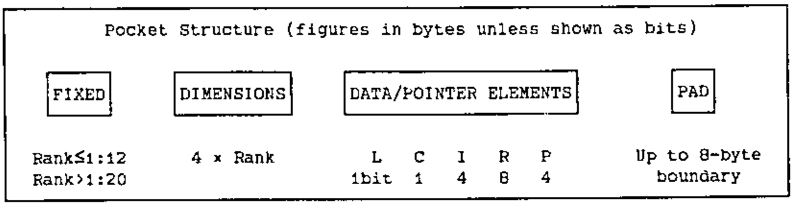
```
⍎,' ',1⌿VARS
```

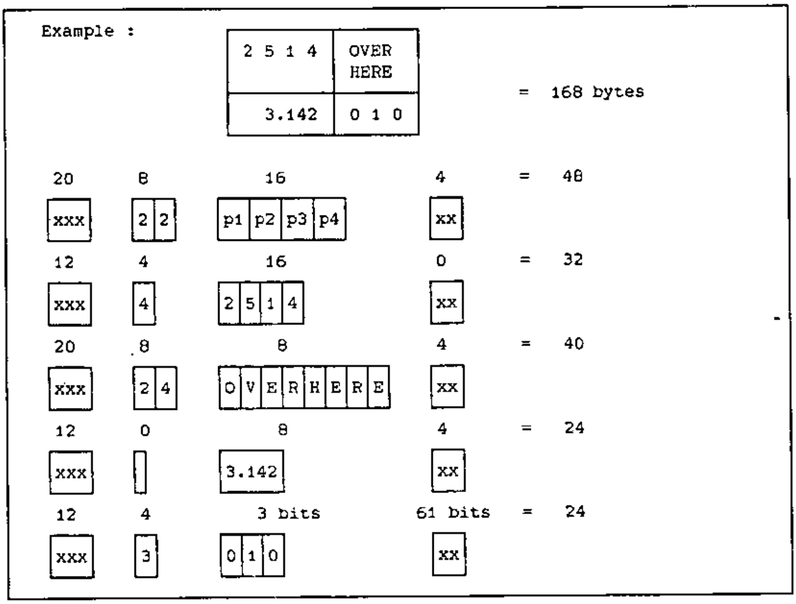
where VARS is a matrix of variable names.

Storage Of Arrays

When designing applications using nested arrays, it is very important to be aware of the workspace implications. Even with IBM mega-workspace, it is all too easy to gobble bytes on a prodigious scale in the heady early days of using nested arrays. The nesting overhead for non-simple arrays can be high. The size figures in the examples shown below apply to APL*PLUS, but other implementations are likely to be of similar magnitude.

The unit of storage in arrays is called the POCKET. The data/pointer elements of a pocket are homogeneous, either all data of one type or all pointers. In a simple, homogeneous array, there is a single pocket. In any other array, there is more than one pocket. (Think of the implications of an array, albeit simple, with both character and numeric elements).

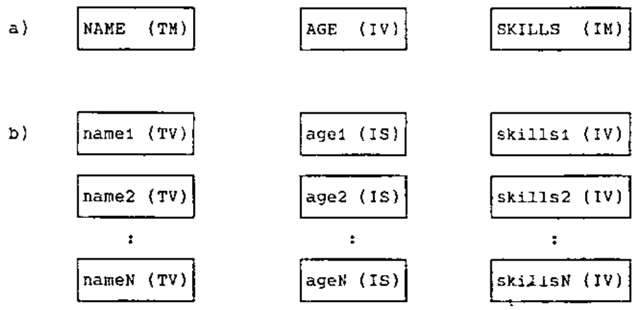




Taking the PERSONNEL example from the previous section, consider the workspace implications of the two structures below with realistic amounts of data:

a) a structure that presents the information in a very convenient form for selection purposes. However, retrieval and updating is not convenient.
b) a structure that is also convenient for selection purposes (since most functions such as epsilon and iota work on nested vectors). Retrieval and updating on a record basis are greatly simplified.

In fact, we see that there is a very large structural overhead incurred with approach (b), whilst that for (a) is trivial. The implications must be carefully assessed before adopting approach (a).



Size information :

	(a)	(b)
NAME	200 × 10	200 × 7 (say)
AGE	200	200
SKILLS	200 × 10	200 × 5 (say)
Data storage	10,800	6,400
Array storage	10,912	16,032
Overheads	112	9,832
Unit increment	56	84

Practical Usages of Nested Arrays

Inverted Data Structures

This usage of nested structures was the subject of the last section. It avoids the need for a multiplicity of global variables that would otherwise be needed to carry information for many fields. Coding is also greatly reduced since many operations that need to be carried out on all fields may be executed in a single expression, eg

```
PERSONNEL←(⊂I)⌿PERSONNEL
```

There is a further advantage on systems liable to interrupts; either all items (fields) are updated, or none are. This same advantage applies when updating a component file since the structure may be stored in a single component instead of one component per field.

Control Blocks

Utility software such as screen-handling or structured file systems often require many miscellaneous items of information to keep track of the environment. A file system may need a tie number (IS), a file-name (TV), an index of field names (VTV), and so forth. Conventionally, such information is stored in an untidy mass of globals — take a look at proprietary screen software such as APE sometime!

All of this information can be carried as a single nested array in a variable whose name is chosen by you as the designer. This variable may then be a common argument to utility functions, and returned, possibly modified, as the explicit result.

Polyadic Functions

Functions suffer the drawback of allowing no more than two arguments. A nested vector allows any number of arrays to be gathered, and the process to be reversed inside the function. For example:

```
      ∇ R←INSERT PARS;I;X
[1]    ⍝ Inserts values into subscripted positions
[2]    ⍝ PARS ←→ <vector to be modified><subscripts><new values>
[3]       R I X ← PARS
[4]       R[I]←X
      ∇
```

Lists

Tables of standard information, such as product names, may be stored as vectors of text vectors (VTVs). Since most standard functions apply to nested arrays, VTVs simplify coding. For example:

```
      PRODUCTS←'HAMMER' 'CHISEL' 'SCREWDRIVER' 'SAW' 'BRADAWL'


      PRODUCTS ⍳ 'CHISEL' 'SAW' 'PLYERS'

2 4 6
```

Operator Extensions

The introduction of nested arrays to APL is, in my view, justified not so much by the added versatility of data structures per se as by the enormously increased richness brought about by operator extensions. A new power has been provided comparable in magnitude to the difference between traditional APL and, say, FORTRAN.

In the past the distinction between operators and functions has, to most practitioners, been fuzzy. Because the number of available operators was small (reduction, scan, inner product, outer product, and, some would say, axis), their slightly different syntax has been considered a special case. In fact, the rules for applying operators have always been clear to theorists (although the ° in outer product is an anomaly). In the true tradition of APL, the rules are simple and consistent. An operator is applied to one or two function (or array) operands to form a DERIVED FUNCTION that is then applied to one or two array arguments. Operators have precedence over functions. They have long left scope (functions have long right scope). That is to say, the left operand of an operator is the longest function expression to its left; its right operand, if there is one, is the function (or array) immediately to its right. The right operand may, of course, be extended using parentheses. For example:

```
      + / 1 2 3             + is the left operand of monadic operator /

6                           +/ is the monadic derived function


      +/¨ (1 2 3) (4 5 6)   +/ is the left operand of monadic operator ¨

6 15                        +/¨ is the monadic derived function
```

The limitation on the extension of operators was the requirement that the result of the derived function be simple. Nested arrays remove this constraint. The re-definition of existing operators and the newly-introduced operators vary between the different implementations. Sharp APL has a number of operators of great power but restricted to a defined set of primitive functions. Other implementations have introduced operators with simpler definitions applicable to any function (including user-defined functions) meeting valency requirements. APL2 even permits user-defined operators. The further discussions in this paper follow the APL*PLUS model.

Extension of Existing Operators

Existing operators are re-defined to work on nested arrays. Taking "reduction" as an example, the new definition for the vector case is:

```
          +/ e1 e2 e3  ↔  ⊂(⊃e1) + (⊃e2) + (⊃e3)
```

whence:

```
                      +/ (1 2 3) (4 5 6) (7 8 9)

                ┌────────┐
                │12 15 18│
                └────────┘

                      ,/ 'ABC' 'DEF'

                ┌──────┐
                │ABCDEF│
                └──────┘
```

A user-defined function may be written to exmine the way an operator works:

```
                       ∇ R ← A PLUS B
                   [1]   A '+' B 'is' (R←A+B)
                       ∇

                      PLUS/ (1 2 3) (4 5 6) (7 8 9)

                4 5 6 +  7  8  9 is 11 13 15
                1 2 3 + 11 13 15 is 12 15 18
                ┌────────┐
                │12 15 18│
                └────────┘
```

Inner product is an interesting operator to examine using this technique.

New Operators

APL*PLUS introduced a single new operator of great power called "each" written as ¨. It is a monadic operator that may be applied to either a monadic or dyadic function to produce a monadic or dyadic derived function respectively. The derived function is applied to each (corresponding) item of the argument(s) after scalar extension if required.

```
                  ⌽¨ (1 2 3) (4  (5 6))

             ┌─────┐ ┌─────────┐
             │3 2 1│ │┌───┐    │
             └─────┘ ││5 6│  4 │
                     │└───┘    │
                     └─────────┘

                  6↑¨ 'DAVID' 'BILL' 'ME'

             ┌──────┐ ┌──────┐ ┌──────┐
             │DAVID │ │BILL  │ │ME    │
             └──────┘ └──────┘ └──────┘
```

The great potential for "each" is to be found when applied with user-defined functions:

```
          TRIM '   some  text    '

      some text

          TRIM¨ '  Hammer   ' '     Monkey   wrench ' '  Saw '

      ┌──────┐ ┌──────────────┐ ┌───┐
      │Hammer│ │Monkey wrench │ │Saw│
      └──────┘ └──────────────┘ └───┘
```

Summary

Operator extensions made possible by nested arrays undoubtedly improve the modularity of APL. This is seen in the ability to write more compact code. The need for looping is all but eliminated. These advantages should lead to more readable, more maintainable code.

There are pitfalls to avoid. Greater power in the language presents many more ways of doing the same thing badly. A common fault when first trying the new facilities is to over-use the "each" operator — I have seen 6 or more occurrences in a single statement. Examination almost always suggests a more succinct approach by collecting the functions involved into a user-defined function that is then applied with a single "each", eg

```
          ⊃¨ρ¨ρ¨ARRAY       is replaced by       RANK¨ARRAY
```

This technique also avoids large temporary usage of the workspace. Consider, for example, the implications of:

```
                        ρ¨⎕FREAD¨TIE,¨COMPTS
```

Deeply-nested structures should also be avoided. Apart from the quite considerable structural overhead in workspace, items that are deeply-nested are difficult to manipulate often leading to over-complex code.

A word must be said about computing efficiency. We are currently seeing first generation implementations of nested arrays, although second generations are now emerging. The nested approach may be slower than a non-nested solution to the same problem. This may be because the primitive functions and operators are not so slick with nested structures, or it may be because the approach per se demands far greater use of resources. The former reason should improve as implementations are enhanced (we are already seeing this). Otherwise the computational cost is a factor to be traded-off against the benefits of programming ease and maintainability.

I have found working with nested array versions of APL a delight. With awareness the pitfalls are easy to avoid. Going back to 'ordinary' APL reminds me of a time some while ago when I was required to program again in FORTRAN; perhaps a little extreme as an analogy but nested-APLers will know what I mean.

APL People Ltd.

17 Barton Street,
Bath,
Avon,
BA1 1HQ
Tel: 0225 62601

APPOINTMENTS

A placement service solely for the APL community.

Use our extensive knowledge, contacts and experience in the APL world.

We have vacancies NOW, both permanent and contract .... and our confidential advice on career moves is always freely available.

CONSULTANCY

A consultancy service designed to solve your APL problems.

Consultant's Toolkit and Supervision available.

Current projects include:-
* graphics
* project management
* databases
* system design
* system 'rescue'
* documentation
* VSAPL under VSPC, CMS, TSO
  APL*Plus/PC, Sharp APL

To discuss your requirements: — Call Valerie Lusmore on 04626-77375

Registered employment agency licence SE6440

H.M.W. PROGRAMMING CONSULTANTS LTD

Why not discover more about

- ★ **Consultancy/Support Service**
- ★ **APL on the IBM PC**
- ★ **VM/CMS Packages**

Ring Ken Jackson at

H.M.W. PROGRAMMING CONSULTANTS LTD

142 FELTHAM HILL ROAD,
ASHFORD, MIDDLESEX TW15 1HN
Telephone: Ashford 41232

Mainframes, Micros and Co-existence
May 28th 1985

The Royal Over-Seas League

by Adrian Smith

Introduction

Here we had two papers along very much the same lines. Graham Fieldsend and Dave Chivers both explored different aspects of the problem of sharing data and workload between mainframes and PCs.

Graham Fieldsend (Tesco) set out the basic problem. They have APL*PLUS on both mainframe and PC, and they find that:

- users only want one terminal
- they want to get at mainframe data on their own machines
- they need to use APL systems downloaded from the mainframe.

By installing the IRMA board in the PC, and acquiring a good deal of ancillary software, they can answer most of these needs. However the installation was not without problems, and downloading APL workspaces is not straightforward. There is also a problem with the timings, as it can take several hours to ship a 1 megabyte file through standard IRMA (there are quicker ways of doing it however) and around 2 hours to reconstitute an APL workspace (100K) once it has arrived.

Dave Chivers talked more in terms of local networks with a single gateway to the outside world. These tend to be high bandwidth (hence very fast) as long as you are talking about PCs on a single site. In his view local processing will continue to increase in importance and the need for remote processing will decline except for occasional access to data bases.

Networks are getting faster, cleverer, more secure and more flexible year by year. Soon the network will itself be able to dial up, sign on, and fetch data from a remote database, all quite transparently to the user. However there are some snags to watch for: errors become much more important when vast amounts of data are automatically getting booted around the place; security is obviously a big headache; modems need to get a lot friendlier. However in spite of these worries, Dave was in little doubt that the future will see a significant move towards LANs and Gateways.

The discussion centred round the problems of getting all this hardware and software to co-operate on a variety of different combinations of mainframes, operating systems and PCs. There was also the question of cost effectiveness: are users shipping data up to PCs simply because they already know and understand Lotus? Would we do better to keep the processing close to the data source by providing better mainframe software?

Finally what about those amazing compact discs? Is the GPO an adequate network when you can ship Gigabytes per day for the cost of a first-class stamp? The answer appears to be 'yes' as long as you don't need to cross an international boundary. Customs folk happily ignore data travelling down wires, but tend to be upset by the same data in visible form. Pity.

MY FIRST DATE WITH IRMA

by Graham Fieldsend

Tesco has invested heavily in IBM hardware, both in mainframes and micros. At present we have twin sites connected by a megastream (see fig. 1). In addition we have over 200 IBM PCs distributed throughout the two sites. We run APL*PLUS under TSO on our Cheshunt mainframe, and we have a number of PCs running APL*PLUS/PC.

Such a set-up led to three problems which we needed to resolve. Firstly, it caused a proliferation of terminals, good for IBM no doubt, but not so for me as I only have a small desk. Secondly, we needed to transfer data between mainframe and PC. Thirdly, we wanted to be able to transfer APL systems between mainframe and PC.

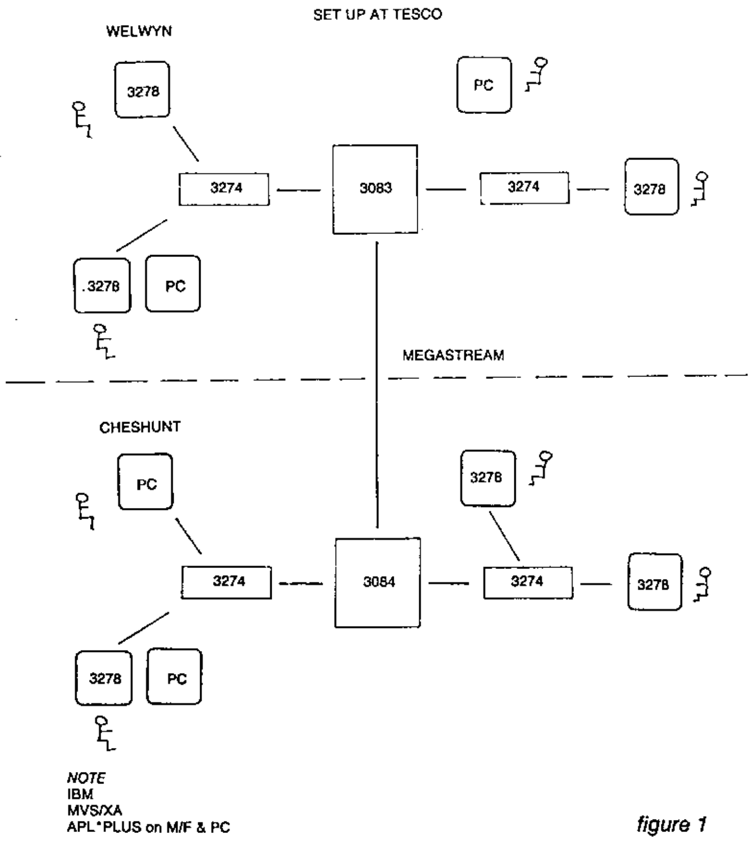


figure 1

Our solution was to purchase IRMA, along with APL*PLUS/PC TOOLS, at a cost of about £900 and £300 respectively.

IRMA is a communications link between an IBM PC XT and an IBM 3270 network. It consists of four parts (see fig. 2).

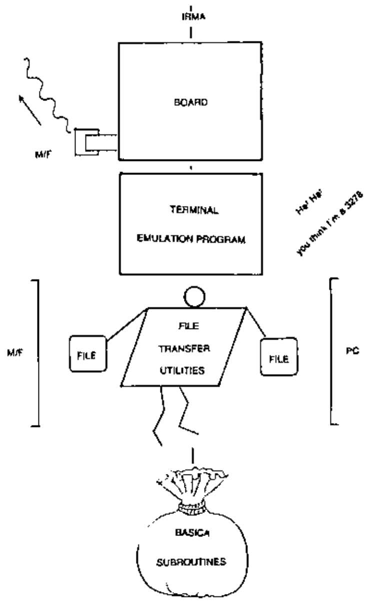


figure 2

The Decision Support Interface (DSI) is a printed circuit board. It plugs into the PC system unit and has a BNC connector for a coaxial link to a 3274 or 3276 controller. It has its own microprocessor to handle the 327X protocol, which is independent from the PC's 8088 chip, and this starts accepting 327X protocol as soon as the board is installed and receiving power.

The Terminal Emulator Program (TEP) is needed in order to see the information received by the DSI, and send keystrokes made by the operator. As its name suggests, it emulates a 3278 screen (with a few additional features).It is very easy to invoke and may be left and reentered at will. An additional program called GENX allows customisation of the program with such features as APL keyboard.

The BASICA subroutines are provided for BASIC programmers who wish to write their own transfer programs. Also, supplied utilities allow easy transfer of files between mainframe and PC, via the TSO editor and IRMA screen buffer.

At Tesco we actually use a new, faster file transfer utility called IRMAlink. It is simple to use, requiring the READY prompt on the mainframe, and providing a fullscreen menu to prompt for file names. A number of different file types may be transferred, such as CLIST, CNTL, TEXT and DATA.

IRMA is able to solve our first two problems, allowing us to use the TEP to emulate a mainframe terminal and the IRMAlink to transfer data files between mainframe and PC. The third problem, transferring APL systems, was covered by TOOLS.

The IRMA software contained in the TOOLS package divides into sections that equate with the IRMA supplied software. TOOLS has a customised version of the TEP, with APL*PLUS/PC layout. Its customisation is such that it cannot be reentered as quickly or easily as the standard version, but these problems can largely be overcome by using a RAM disc and defining a function key.

The functions in the I78FNS workspace parallel the BASICA subroutines (with a few added extras), allowing APL programmers to write their own transfer programmers.

The file transfer utilities are contained in the I78XFER workspace, and work in conjunction with the I78HOST workspace, which must first be installed on the mainframe. This installation is not easy, to say the least. Local differences at each site, such as the position of the characters 'APL' on the screen, and the type of hardware between the 3278 and mainframe, can make this difficult if not impossible. In our experience, once installed it is slow but reasonably reliable. Several types of file may be transferred, including Shared Files, and functions are provided to store a copy of a workspace on file and restore it after transfer. GDDM must be installed on the mainframe, as this is used to mimic quad-ARBIN, which is used at the PC end. Character vector representations of data are transferred between mainframe and PC, with a quad- AV conversion taking place.

So the TOOLS package was used to solve our third problem. Finally, I supply some timings as recorded on our mainframe during a normal working day. The timings for IRMA and FORTE are for the transfer of a file of total size 1 megabyte, and record length 255 bytes:-

	Download	*Upload*
IRMA	1hr 58min 31sec	4hrs 39min 48sec
FORTE	42min 8sec	47min 1sec
IRMAlink	6min 4sec	6min 40sec

Transfer of a 100K workspace using the TOOLS IRMA based software took about 2 hours.

APL STATISTICS USER GROUP

The re-formed group met in the pleasant surroundings of the Customer Centre, IBM, South Bank, on 5th June, at 12.30 p.m. After lunch, overlooking the Thames, talks were given by Professor Barry Wetherill and Professor Alan Hawkes.

Barry Wetherill spoke on "Intelligent Statistical Software", illustrating initially how "dumb" software failed to highlight problems in data sets. For these various problems he suggested appropriate diagnostic tests which could be used. He concluded by showing how the measures are included in the Survey Package he designed including data validation.

The second talk by Alan Hawkes was on the "Role of APL in Teaching and Research". This talk detailed the approach used at Swansea to teach statistics students through the medium of APL. He then illustrated his own use of APL in research emphasising the speed of using APL compared with other languages.

After the talks a discussion of the future role of the group took place, and format of its future meetings.

One suggestion which arose was the formation of a Program Subgroup to consider the possibilities of a standardized APL workspace for statistics. The group were not totally in agreement over the role of this subgroup, but thought it worthy of pursuit.

Another suggestion was the meeting should allow for "hands-on" experience of software. It was thought that future meetings would have a workshop session in the morning followed by talks in the afternoon. To resolve other problems it was decided to send out a survey! For further details of either program subgroup or future meetings of group, please contact:

Jake Ansell,
APL Statistics User Group,
Dept. of Management Science & Statistics,
University College of Swansea,
Singleton Park,
Swansea,
SA2 8PP.

APL85 — A BRIEF IMPRESSION

by Dick Bowman

Unfortunately this is a rather briefer overview than I might have hoped to achieve, promoting APL86 taking somewhat more time and attention than I had anticipated. Having heard some rather disheartening rumours about the pre-event registration numbers I was ultimately pleased to find myself in the company of over 450 other delegates.

The Setting

Seattle is a greatly underrated city. Aside from registering in my mind as the venue for one cataclysmic event nearly twenty years previously, I knew absolutely nothing beyond roughly where it was and that it was somewhere I definitely didn't plan to visit. (Bob Gailer's slide show at APL84 gave me some glimmering that I might have got hold of a slightly poor impression). Nevertheless, APL wins, and the Big Iron Bird does its bit; at this point, an aside to those who might follow — Seattle airport has the most outrageously complicated system of baggage handling. You are not recommended to consume your entire duty-free allowance on the flight unless you have a well-developed faith in the honesty of your fellow man. But we only lost one box of Vectors, which turned up the next day.

After staggering into the official British APL Association accommodations (why wasn't the Y good enough for everybody else, I ask?) we awaited the dawn — well, actually we awaited the sunset as well. Another of Seattle's less documented attractions being the Allnight Demolition Company, who set about the task of knocking down an entire block while we were there.

But beyond that the overall impression was of a pretty pleasant place to be. You had to be fairly determined to get run over; we didn't get mugged; we had some pleasant meals (I recommend F X McRory's and Enoteca); the setting is splendid (aside from the work of the idiot who built a two-tier freeway right along the waterfront); and above all you knew where you were (Seattle is not a faceless tourist trap).

The Conference

A quick scan through the papers had revealed that this was to be a conference which looked rather more outward from the introspection of so much APL; the balance was good. As is always the way there was more of interest than a single person could attend; it seems invidious to pick on individual papers for comment, but nevertheless I'd like to make one or two observations in certain areas (if my conclusions differ from those of the original authors I claim poetic licence).

APL and Graphics:

Having long been of the opinion that the two were made for each other I found papers by Roos/Laitonern and Ylinen on the practicalities of developing specific business-oriented graphics interesting. With a common theme that simple things are worth doing (because they'll grow into bigger and better things), the former dealt with the iconic approach to presentation building, the latter with how easy it is to get started on a presentation graphics package. (Would these guys like to borrow some users? I have a few spare.) Also a more fundamentalist message from Richard Nabavi toward the goal of aligning ourselves with the graphics world, paralleled by Neurdenberg and Schwarz in their selection of GKS as an outside system to interface with APL. The pre-conference seminar on the subject of 3D graphics was somewhat tangential in its relationship of the subject matter to the conference proper.

Promoting APL:

The opening address by Adin Falkoff delved back toward the earliest days of APL conferences and the 'wish-list' of the delegates of those times. So much has been answered, one could conclude that APL is now well positioned to move out of its minority placing.

Myrna and DiChellis presented a statistical survey giving some evidence that, for at least one important category of computer user, the picture of APL as declining minority interest is somewhat removed from the truth — BASIC does not rule the world. It would be interesting to see this survey widened.

Peele/Eisenberg gave a thoughtful insight into that most important aspect of promoting APL, which is teaching it to the newcomer (for me a little too high on the questions and the negative rules and a little light on positiveness).

A lunchtime talk by Tama Traberman (enlivened on at least one table by a waiter who could teach Basil Fawlty a trick or two) brought us up to date on her APL83 paper and gave reinforcing evidence that APL has its attractions as a part of schoolroom education.

Another aspect of APL which is coming to be more significant is its use as a development language for packages which are perceived by end users as nothing other than a solution to a problem, the user isn't sold APL; many of us have, of course, been doing just this for some time but we are now also beginning to see the emergence of marketplace software for which APL has been crucial in development but in which APL is not made evident, Kevin Weaver gave an insight into the advantages of APL in this arena and into the approaches for successful marketing. Combine this with the practical guidance on software publishing given by Philip Evans.

Always at these events Paul Berry's contribution is a thoughtful one; some may think that his APL85 paper is more tangential to the theme of promoting APL than I do. Paraphrasing frantically I received a message that the original layout of the APL keyboard was misguided in its moving of established symbols but that the adoption of a symbolic system moved APL away from and beyond pseudo-English with its ambiguity and cultural aloofness — we could do well to change the keyboard but making APL a keyword language might not be a totally clever idea.

APL for the APLer:

Soop/Davis made some eminently sensible suggestions about expanding the scope of shared variables (persistence between sessions and explicit rejection of shares); another paper which caught my particular interest was by Tom Pritchard with a suggestion for an APL macro facility which leads to many seeming simplifications in code as written as part of an application.

My main regret was that I was unable to attend any of the tutorials or much of any of the panel discussions, which is a pity as these are just the things which usually fail to be documented in the proceedings. Agreed, there was a gentleman selling audio tapes of many of the sessions but I don't find that audio is an appropriate medium for APL (also as a poverty-struck Briton I felt that conference documentation should not be additional to the conference fee).

The Exhibition Staged on a lesser scale than one might have expected and with one or two conspicuous absences. Of course one admires the foresight of Dyadic Systems in having a stand prepared in every continent and the sheer battleship majesty of the Analogic booth. Two products to particularly catch my eye were the Ampere machine which distinctly falls into the category of the machine I'd most like Father Christmas to bring me, and STSC's Pocket APL which joins QL/APL in being an implementation at an individually-affordable price. (I particularly approve of STSC's slogan.)

Overall

Notwithstanding the pre-event whisperings at the end of the day we should recognise that this was a technically successful and entertaining conference. The significant amount of work put in by Bob Gailer and his committee made it a relaxed event. In particular Kevin Weaver's obvious energy kept things on the move, and Harvey Krilloff and Joel Ware helped make our presence an active one.

APL and the Future; well, the future's APL86.

REPORT FROM APL85 — APL AND THE FUTURE

by John Adams

The conference spanned 4 days with Tuesday and Wednesday by far the most hectic. By bag-on-a-string-making-weak-brown-liquid-time I found I was saturated with information and felt like doing nothing but settling down with a nice steaming pot of tea for the rest of the day. But this was not to be. On Tuesday the conference committee had organised an evening boat trip across the Puget Sound with food and entertainment in the Indian village of Tillicum. And on Wednesday was the conference banquet including a very humorous talk by Andrew Tobias. Since the talk was basically how to make money playing the stock market it went down particularly well with the majority of the audience.

On the serious side there were also tutorials, panel discussions, open discussions, an exhibition, and vendor's presentations. The tutorials and panel discussions worked particularly well. The exhibition was more variable: just about everyone was there, including our own association selling copies of Vector like hot cakes (or potatoes?); STSC, Dyadic, and Analogic had the most impressive implementations all winning prizes in the computer-off competitions; IBM's stand was rather disappointing and seemed to lack the commitment of others; IP Sharp's main distinction was their complete absence from the exhibition (our loss or their loss?).

The conference spanned 4 days with Tuesday and Wednesday by far the most hectic. By bag-on-a-string-making-weak-brown-liquid-time I found I was saturated with information and felt like doing nothing but settling down with a nice steaming pot of tea for the rest of the day. But this was not to be. On Tuesday the conference committee had organised an evening boat trip across the Puget Sound with food and entertainment in the Indian village of Tillicum. And on Wednesday was the conference banquet including a very humorous talk by Andrew Tobias. Since the talk was basically how to make money playing the stock market it went down particularly well with the majority of the audience.

For me the conference was a wonderful blend of going to new places, expensive breakfasts, APL, new friends, Space Needles, mountains, APL, clam chowders, 'plane rides, APL, jig-saws, trees, APL, monorails, APL, and more APL. All of those at APL85 were looking forward to APL86 and meeting up with old and new friends in Manchester. I hope I can be there and hope you can make it too.

The conference papers were quite varied and yet one or two topics seemed to keep cropping up: viz education, teaching, and learning; APL2; and efficiency of APL (all of which are important aspects for the future of the language). It is difficult to recommend particular papers since their interest will depend on why you are interested in APL. Some of them may seem irrelevant to the environment you are working in. Nevertheless the papers on GKS, Teaching 'Bugs', Improving Performance with APs, Operators for Program Control, and the Introduction to STSC's Compiler are especially worth reading by anyone with a view to APL and the future.

APL85 — Second Thoughts on Seattle

by Dick Gray

My overwhelming impressions of Seattle are its beautiful landscapes and seascapes set against the mountainous backcloth of the Rockies, and the immense exuberance, gaiety and friendliness of the American people, particularly those who are fortunate enough to live on the West Coast. This exuberance may be a result of the obvious signs of affluence all round — the well-heeled commuters on the smart ferry to Winslow, the chic department stores and the articulate, fresh-complexioned college students who are everywhere to be found 'moonlighting' as barmen, waitresses, cashiers and counter assistants.

But on a more sombre note, Japanese cars, goods and salesmen are evident everywhere, and one cannot help but wonder whether this could be the cloud about to cast its shadow over such obvious prosperity.

'Hi-tech' is the order of the day. Everything seems to open and shut, turn off and on, and go up and down without even the assistance of the human hand or foot. Magic is indeed in the air — no wonder they have to go jogging for exercise? But I mustn't knock it. That was the voice of envy speaking.

And now I come to eating. No envy here. American food is, we are assured, wholesome; but first one needs plenty of time to read and translate all the 'hype' on the menu, after which I found the actuality somewhat disappointing — all presentation and little flavour, especially the inevitable clinical side-salad of "fresh hand-picked crispy lettuce, fine shredded carrot and polished button mushrooms". (How else can one pick lettuce?). Only breakfast lives up to the menu although the size of my pancake 'stack' had to be reduced throughout the four days on an exponential scale (4,2,1,0).

Which brings me to the APL85 Conference itself. Like the food, it was slick and well presented, but I found a lack of originality in many of the presentations; perhaps I was just unlucky in my selections, but the interesting case study by Stephen Jaffe of Mobil was really an update of the progress of the Mobil project formerly presented at APL83 in Washington. The lunchtime talk by Tama Traberman was for me the outstanding talk of the conference, but I seem to remember her too presenting something very similar at APL83.

And a final comment — I can't bear to be lectured to whilst I am eating; this also applies to the banquet where our table was 'beyond the Pale' and we certainly could not hear enough over the clatter of service to understand why anyone else was laughing. Organisers of APL86 take heed!

Since Dick Gray Associates is right up-to-date in at least one important aspect of computer technology — we operate from an office at home in the heart of the Leicestershire countryside — I attend these conferences with the main aim of keeping in touch with APL events and people. For me APL85 was good on people, less good on events.

JAPANESE APL ON IBM 5550

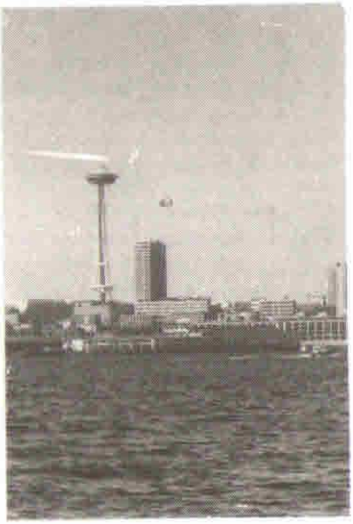
One of the more interesting displays at the APL85 exhibition was Japanese APL on the IBM 5550, apparently a mutant IBM PC. The Japanese characters (Kanji, Hirakana or Katakana) are stored internally as a new data type, with two bytes per element, occupying two columns when printed or displayed. They can be mixed with conventional one-byte characters, and can be processed via APL primitives (rho, transpose, indexing etc.) as normal.

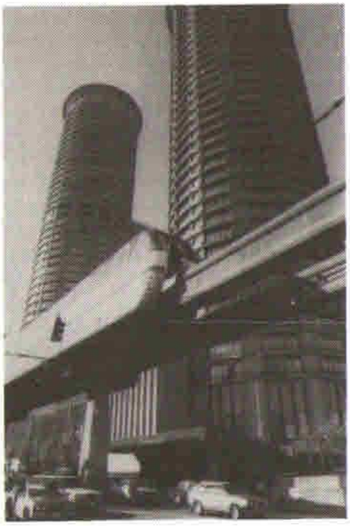
The product was produced jointly by the IBM scientific centres at Tokyo and Madrid.

```
┌──────────────────────────────────────┐
│  IBM Multistation 5550 日本語 A P L   │
│            Version  1.01             │
│  (C) Copyright IBM Corp. 1984, 1985  │
└──────────────────────────────────────┘

             Produced by
      IBM Tokyo  Scientific Center
      IBM Madrid Scientific Center

CLEAR WS
      City←'東京　名古屋大阪　長崎　'
      ⍴City
12
      City←4 3⍴City
      City
東京
名古屋
大阪
長崎
      ⌽City
　京東
屋古名
　阪大
　崎長
      ⍉City
東名大長
京古阪崎
　屋
      Yomi←'とうきょうなごや　　おおさか　'
      Yomi←Yomi,'ながさき　'
      ⍴Yomi
20
      ⌽Yomi
　きさがな　かさおお　　やごなうょきうと
      Yomi←4 5⍴Yomi
      Yomi
とうきょう
なごや
おおさか
ながさき
      Data←?4 3⍴9999
      Data
 1316 7556 4587
 5328 2190  471
 6788 6793 9346
 3835 5194 8309
      City,'   ¥555,553'⍕Data
東京          ¥1,316        ¥7,556        ¥4,587
名古屋        ¥5,328        ¥2,190          ¥471
大阪          ¥6,788        ¥6,793        ¥9,346
長崎          ¥3,835        ¥5,194        ¥8,309
```

(Continued)

```
      City∧.='大阪　'
0 0 1 0
      (City∧.='大阪　')⍳1
3
      Data[(City∧.='大阪　')⍳1;]
6788 6793 9346
      ∇a←Lookup p
[1]  a←Data[(City∧.=p)⍳1;]
[2]  ∇
      Lookup '長崎　'
3835 5194 8309
      +/Lookup '長崎　'
17338
      ⍴AIUEO
55
      ⌽⍉11 5⍴AIUEO
こんるやまはなたさかあ
ゃがれゆみひにちしきい
ゅぎろよむふぬつすくう
ょぐわらめへねてせけえ
っげをりもほのとそこお
      AIUEO⍋Yomi
3 1 4 2
      City[AIUEO⍋Yomi;]
大阪
東京
長崎
名古屋
      ∇Report;inx;p;d
[1]   inx←AIUEO⍋Yomi
[2]   p←City[inx;]  ⍝都市名を読みでソート
[3]   d←Data[inx;]  ⍝対応するデータもソート
[4]   ⎕←p,'   ¥555,553'⍕d
[5]   ⎕←''
[6]   ⎕←'合計   ','   ¥555,553'⍕+/[1]d
[7]   ∇
      Report
大阪          ¥6,788        ¥6,793        ¥9,346
東京          ¥1,316        ¥7,556        ¥4,587
長崎          ¥3,835        ¥5,194        ¥8,309
名古屋        ¥5,328        ¥2,190          ¥471

合計         ¥17,267       ¥21,733       ¥22,713
```

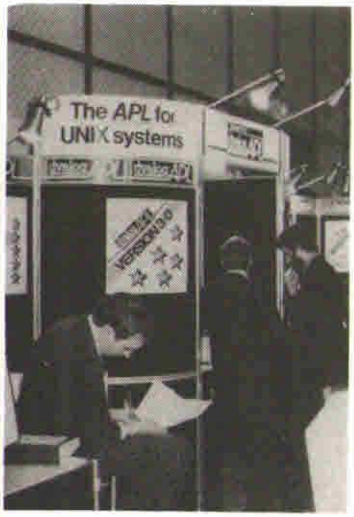




APL 85 took place in Seattle, Washington (for those with bad geography, Seattle is to be found near the top left on a map of the USA!). The venue was the Westin Hotel (left), with its imposing twin towers (a great location for a conference but too expensive to actually stay there). The monorail in the picture takes you to the Space Needle (above).

"Look no hands!" Jim Ryan demonstrates the super-fast Analogic array processor.

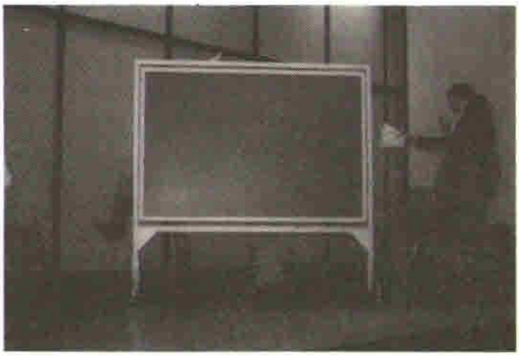
James Wheeler shows off APL*PLUS/UNX running multiple APL sessions on a Sun workstation.



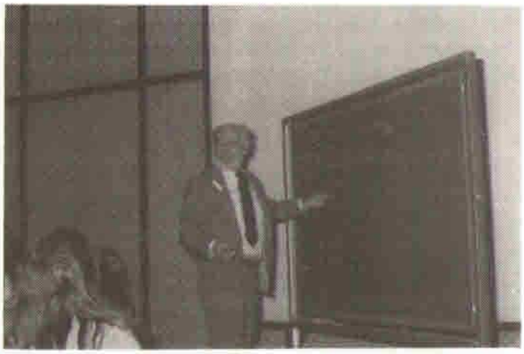
Each APL vendor at APL85 was asked to solve problems set by the other vendors. Here John Scholes puts the finishing touch to one of his answers.

One of the entertainment high spots must surely have been the excellent Kutamba band who played for us after the boat trip across the Puget Sound to Blake Island.

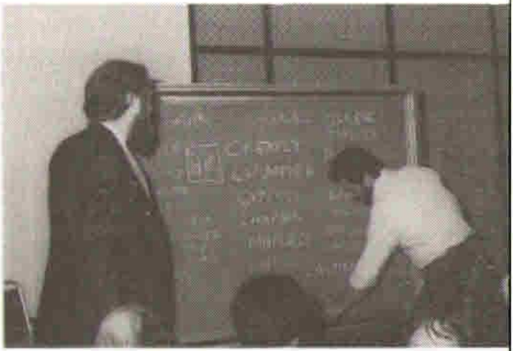
Bob Gailer, APL85 Conference Chairman, tries out one of the new knee-top micros, and the Program Chairman clearly can't believe his eyes.



Roy Sykes (on the left with a stack full and on the right with a clear workspace), gives a welcome repeat of his famous APL81 cabaret. Also on Banquet Night a rendition of 'APL Blossom Time' was given with a few new verses added (this song is an amusing and often historical account of the development of APL).



Banquet fun: everyone was split into three teams and pitted against each other in a variety of ways. The photographs above show two of the teams trying to find as many words as possible which contain the letters APL in the correct order. Adin Falkoff (top) looks for help from his team in adding a few more words to their list. (Adin, as it turned out in another game, can write backwards - not just in capitals but in script - very useful for coding APL presumably).
Underneath, Kevin Weaver counts up the number of correct words for Crick's team; "Chapel" and "Alpine" are definitely suspect but "Cheaply" and "Haploid" are great!

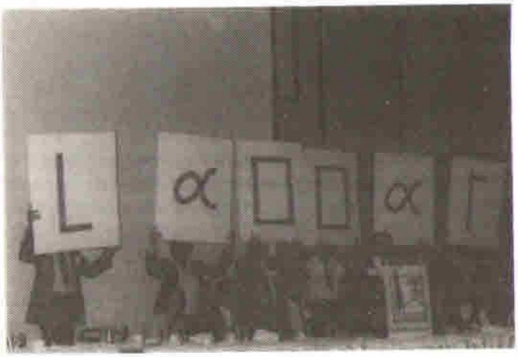


If you've ever wondered what it's like to be a cell in an array then this game is for you - but you'd better be ready to move to the right place when someone shouts out "two three reshape" or "one rotate in the first direction

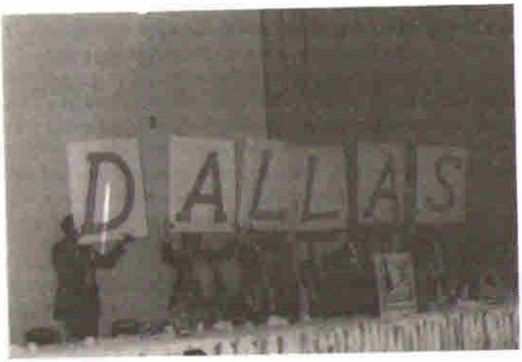
Ian Sharp and Dan Dyer (at the rostrum) receive their outstanding achievement to APL awards.

Philip Goacher tells an excited audience about APL86; there was much applause when Phil explained that software demonstrations would accompany the delivered papers at APL86.

Invited speaker Andrew Tobias tells the banquet audience "How to get by on $100,000 a year (and other sad tales)".



John Carpenter goes to great lengths getting the venue of APL87 across (thank goodness they brought those cards out in the right order).



Fitness consultant Denise Austin gets the packed hall to its feet for an example "Tone-up at the Terminal" session. Everyone present stood up to win the 'Star Prize' - a free Ampere lap-top APL computer.

GENERAL ARTICLES

This section of VECTOR is oriented towards readers who may neither know APL nor may be interested in learning it. However, we hope that you are curious about why, under the right conditions, such impressive results can emerge so quickly from APL programmers.

AN "APL" FOR THE TEACHER

by Dick Gray

It is generally agreed amongst APL 'cognoscenti' that the language has failed to be accepted by the educational and business establishments of the UK to anything like the extent of its promise. At one APL Conference after another, this lack of appreciation has been bemoaned and balefully ascribed to every imaginable reason from the stupidity of "homo sapiens" (the user) to the intractability of God (Ken Iverson).

It is, of course, relatively easy to harp upon this disappointing performance; what is more difficult, however, is to plot a course which will cause the obvious virtues and potential of APL to be better understood by the public.

The purpose of this paper is to propose a line of approach that may enable the objections of the unbelievers to be outflanked and overcome. If this is agreed, the next step would be to decide upon suitable ways of getting the message over to the educators and the users.

When comparisons are made between APL and other languages, the relative merits of each — on performance, conciseness, ease of development, etc. — are usually related as though the purpose of APL was the same as the purpose of other languages. Thus we are accustomed to hearing such remarks about APL as "It is good for the one-off" or "It is just an intellectual exercise for the Great Unwashed." In my opinion we should refer to and, indeed, teach APL as a different species, a language of control; moreover, at present and for the time being **the** language of control.

Consider, if you will, the characteristics of an aeroplane. It has its engines which provide the power, and alongside them there is a complementary system which monitors and checks the engine performance and fuel supply, namely the control system. This symbiosis of ordered power is closely inter-connected through a network of nervous system forming a coherent whole, but the two elements are very different in function and purpose.

A company operates in a very similar way. It needs a high-volume DP system to "drive" the organisation (sales accounting, payroll, ledgers, etc.) but it also needs a planning and control system with entirely different attributes to the "heavy-duty" characteristics of transaction-based DP. By contrast, this system has low data volumes and is fed with summarised, rather than detailed information.

I do not subscribe to the idea that this latter control system should form part of the "main-drive" system any more than that the mind of a human being should be mixed up with the locomotory muscles. If the Almighty had wanted it that way surely he would have put our brains in our bottoms.

I propose therefore that those people who design computer systems, or who teach computer studies, should be encouraged to think about installations in two separate parts, the main processing "body" and the control system or "brain". Computer managers would not have to select one language to support an installation, but two, one of which would automatically be APL. In this way the great competitive language argument would not arise.

Since planning is an art, and data processing is but a rather pedestrian mechanised science, I feel that the idea of a control language would apeal to the more imaginative and creative recruits, and that general recognition of a need for an installation "brain" would ensure the future place and progress of APL.

ANNOUNCING...

MERCIA
Software Limited

A NEW NAME IN THE APL WORLD

Mercia Software was formed by ex APL* PLUS Ltd. employees, and specialises in APL software, services, consultancy and training.
We are dealers for STSC's range of APL* PLUS micro products:

APL* PLUS/PC
STATGRAPHICS
TOOLS VOL 1 and VOL 2
Financial and Statistical Library

PLUS

★ **STSC's and Parallax Systems Mainframe Information Centre Products.**
★ **STSC's APL* PLUS Timesharing Service.**
★ **APL Consultancy.**
★ **On and Off-site VSAPL and APL* PLUS Training:**

Next 'INTRODUCTION to APL* PLUS/PC' public course
September 25th, 26th, 27th, Birmingham.

★ Write, or call Mercia Software ★

Aston Science Park, Love Lane, Birmingham B7 4BJ.
'phone 021-359 5096. Telex: 334535.

STEPS TOWARDS A BETTER BASIC — Part 4

by Anthony Camacho

This issue we include a further extract from the series in Datalink in which Anthony Camacho attempts to explain some APL concepts in terms familiar to BASIC programmers. Here he shows how random numbers can be put to use.

The article is reproduced by kind permission of Datalink magazine.

The die is cast: the cards are dealt

APL has the edge on RND in BASIC

Fig. 1 shows a way of describing a pack of cards in a table of characters using only the 'RESHAPE' and 'CATENATE' functions. In APL reshape is the Greek 'R' which looks like a small p and catenate is the comma. The table shown has 52 spaces in the first row, more spaces but with the 1 of the 10 for each of the ten-spot cards in the second row, four sets of card names (zero for each of the 10s) in the third row and thirteen of each of the suit letters in the fourth row. The commas act like BASIC + to join the strings together to make a single string (or vector as it is called in APL) 208 characters long, and the 4 and the 52 reshape it into four rows of 52.

```
      FIG 1
      A←4 52ρ(52ρ' '),(52ρ'    1        '),(52ρ'AKQJ
      098765432'),(13ρ'S'),(13ρ'H'),(13ρ'D'),(13ρ'C'
      )
      A

    1            1            1            1
AKQJ098765432AKQJ098765432AKQJ098765432AKQJ098765432
SSSSSSSSSSSSSHHHHHHHHHHHHHDDDDDDDDDDDDDCCCCCCCCCCCCC
```

Perhaps the table will be easier to read if turned through ninety degrees and put into four columns; but we must have the 1 before the 0 in each 10 so the table has to be reflected as well as turned. The APL symbol which is the circle (called pi times) overstruck with a backslash reflects a table about the diagonal from top left to bottom right. The first line in fig. 2 makes a table of 52 lines of 4 characters. There are as many indices in the square brackets as there are dimensions; they are separated by semicolons. Iota 13 gives the integers from one to 13 so adding constants, lets us pick each of the four suits & make a table of them.

```
      FIG 2
      A←⍉A
      A←A[⍳13;],A[13+⍳13;],A[26+⍳13;],[39+⍳13;]
      A
 AS  AH  AD  AC
 KS  KH  KD  KC
 QS  QH  QD  QC
 JS  JH  JD  JC
10S 10H 10D 10C
 9S  9H  9D  9C
 8S  8H  8D  8C
 7S  7H  7D  7C
 6S  6H  6D  6C
 5S  5H  5D  5C
 4S  4H  4D  4C
 3S  3H  3D  3C
 2S  2H  2D  2C
```

The question mark in APL is the symbol for two functions, 'ROLL' and 'DEAL'. Roll is like the BASIC function RND but only produces integers, whereas RND normally will produce a random number between 0 and 1. Some versions of BASIC have a number in brackets after RND, to vary the function. BBC BASIC is unusual in allowing RND(6) to give a random throw of a die. Most BASICS need 1 + INT(6*RND). The number on the right of the function specifies the number of possibilities; in APL the number on the left of the function (if there is one) specifies how many of those possibilities are to be chosen — or how many of the cards are to be dealt!

In BASIC, to produce a dozen throws of a die we would write:

```
FOR I=1 TO 12:PRINT INT(6*RND);:NEXT I
```

but there is no easy way to avoid picking duplicates. Here is one way to shuffle the alphabet:

```
 10 DIM B(26): B$=""
 20 FOR I=1 TO 26
 30 A=1+INT (26*RND): OK=1
 40 FOR J=0 TO I-1
 50 IF A=B(J) THEN OK=0
 60 NEXT J
 70 IF OK=0 THEN GOTO 30
 80 B(I)=A: B$=B$+CHR$(64+A)
 90 NEXT I
100 PRINT B$: END
```

Fig. 3 shows the APL equivalents.

```
      FIG 3
      ⍝ HERE ARE 12 THROWS OF A DIE
      ?12⍴6
5 6 2 1 4 1 6 3 4 3 2 2
      ⍝ HERE ARE 26 INTEGERS SHUFFLED
      26?26
22 4 16 2 11 8 25 15 12 1 13 9 5 6 20 24 21 10 23 18
       3 17 14 7 19 26
      ⍝ TO CONVERT TO LETTERS ADD 64 TO CONVERT TO A
      SCII AND THEN USE THOSE NUMBERS TO INDEX ITEMS
       OUT OF THE CHARACTER SET WHICH APL HOLDS AS ⎕
      AVI
      ⎕AVI 64+26?26
KUHEXTLVIZPGAFSRQBNWMCODYJ
```

An elegant way of expressing a hand of whist or bridge is as a four by thirteen reshape of a deal of 52 out of 52 cards. Our 52 numbers can easily be translated into the names of the cards by picking them out of the table of names that we made (fig. 1). Note the blank after the semicolon in fig. 4 which takes a whole line of four characters.

```
      FIG 4
      4 13⍴52?52
11 50 27 14 35 28 39 38 30 32 47 15 22
26 13 33 16 37 10 43  4 45 46 42 34 41
49 21 23 36  6 20 12  3  5 24 48 19 25
 8 31 29  7 17 52 18  1 40 51  2 44  9
      13 16⍴A[52?52;]
 KS 10H  JC  9H
 9D  KD  QS  7D
 2S  8H  7S  JS
 5D  2D  5H  2C
 3H  3C  4C  JD
 AH  6H  8S  6C
 QC  2H  8D  4H
 3S  8C 10D  5C
10S  AS  KH  7H
 QH  5S  6S  4D
10C  6D  9S  4S
 9C  KC  JH  7C
 3D  QD  AD  AC
```

All the APL shown so far is commands. Like BASIC it can also be used to write programs. An APL program is made of functions, rather in the way a good BASIC program is made of subroutines (or in BBC BASIC of procedures). But whereas all BASICs allow you to write instructions outside subroutines or procedures, APL does not. All APL instructions are in functions and one function must be the main program and call the others. Here in fig. 5 is an example of an APL function called HAND which produces a set of bridge or whist hands as its result. The upside down delta is the symbol which introduces and ends an APL function, and the assignment to R is the way APL specifies that the function has an explicit result. The variable after the semicolon is localised.

```
FIG 5            ∇HAND[⎕]∇
               ∇ R←HAND ;A
          [1]    A←⍉4 52⍴(52⍴' '),(52⍴'    1         '),(52⍴'AKQ
                 J098765432'),(13⍴'S'),(13⍴'H'),(13⍴'D'),(13⍴'C
                 ')
          [2]   ⍝ [1] PUTS THE NAMES OF THE PACK INTO A
          [3]    R←13 16⍴A[52?52;]
          [4]   ⍝ [3] PICKS THE NAMES ACCORDING TO THE DEAL AND
                  PUTS THEM INTO A TABLE OF FOUR HANDS
               ∇

                 HAND
          10C  2S  8S  5D
           9D  5S  AC  KS
          10S  6S  QH  6C
           QD  2H  5H  3C
           9S  4S  5C  7S
           6H  4D  AD  JH
           7C  3S  9H  7H
           8H  4H  4C  7D
           JS  6D  9C  8C
          10D 10H  QS  3H
           KD  AH  2C  AS
           8D  JC  KH  2D
           3D  KC  JD  QC
```

CASE STUDY

by Adrian Smith

Acknowledgements

I would like thank Pat Wood and John Marchant for taking up so much of their valuable time to show me round, and also for being so willing to explain the history and background of the developments described.

Author's Note

Nowhere in this article will you find so much as a mention of APL. I have deliberately and firmly suppressed any personal tendency to think of the APL alternatives to the system described. I hope that when you first read the study you will try to do the same. Then maybe you could take a more critical perspective, and start thinking along the lines 'I could have done that in half the time with APL*PLUS and a couple of PCs', or alternatively 'I'm glad I wasn't asked to do that in APL!'.

Looking back, it seems to me that the 'package' approach was dead right for the standard data entry and reporting, although you could argue that the package could perfectly well be written in APL! However poor DBASE-II was put through some dreadful hoops to produce things like age pyramids; in fact it was made to tackle all sorts of extremely APL-ish problems, and it made very heavy weather of them.

Surely the conclusion is obvious. By all means let us use DBASE-II (or Datamaster or Delta or whatever) for the things they do best. What we need for APL is a standard set of well-documented reliable 'windows' into all the arcane and incomprehensible file formats used by the 'top-10' packages. Michael Carmichael's paper (VECTOR 1.4) is a good start; if anyone else is beavering away along the same path maybe they would like to add to the collection. There are plenty more VECTORS to come!!

DBASE-II at Bedford School

by Adrian Smith

Introduction

These notes are an attempt to give an overall impression of the way DBASE-II has been used at Bedford School to build up an impressive array of indexing and reporting systems around the basic school records. The detailed operation of the programs is obviously not of particular interest; rather the way the system has grown up, and the mistakes made and experience gained along the way.

Background

Bedford School is one of the top public schools in the country; it has about 1100 boys in total (800 in the upper school, of whom 40% board). It is part of a larger grouping of public schools in Bedford, all administered under the umbrella of the Harpur Trust. Over the course of a year around 300 registrations are taken, and there are clearly some quite complex clerical systems to cope with the 'school list' and to estimate the number of potential entrants in any given year.

In common with most 'big' schools Bedford School's maths department acquired a number of BBC micros to teach computing. These soon developed into a 'computer club' under the leadership of an extremely enthusiastic and able master; as you might expect it wasn't long before this began to produce a whole range of ad hoc programs in BBC Basic to tackle the knottier bits of the existing clerical systems.

So useful did the programs become that one of the BBCs soon migrated down to the school office, and started to become an integral part of the whole administration system. Fortunately, enough of the danger signals were spotted in time, and rather than simply letting the ad hoc system grow out of control, the staff decided to step back and ask for expert advice.

Why use a Database?

Lacking 'experts' of their own, the school sensibly looked elsewhere and retained a consultant to advise on the 'best' way of building a really professional system. The brief covered both hardware and software, and the immediate recommendations were:
- — use Apricots
- — use a database

This obviously implied a complete re-write of all those extremely useful Basic programs, and the return of that BBC to its rightful place in the maths department! Why throw away a perfectly workable system in favour of something unknown and untried? It might be interesting to compare the reasons given at the time with subsequent experience:
- — reliability. There is no doubt that the BBC programs were 'breakable'. One could reasonably expect a proprietary package to be much more robust.
- — compatibility. All the Harpur Trust schools would eventually need some form of computerization; if they were to share the same database management system they would find it much easier to share information.
- — speed of development. Typically you would expect a range of useful macros for screen-editing and report generation. These are the sort of thing which take ages to program if you have to do them yourself.

— maintainability. Databases tend to come with a high level 'structured' language which should generate far more comprehensible code than even the best available Basic.
— wide range of 'standard' reports. Using Basic even the simplest ad hoc report needed a fair amount of programming.
— speed. One would expect a professionally developed software tool to run a good bit faster than a mish-mash of schoolboy code!

There was one clear drawback to the database road; the secretaries had already got well used to the BBC programs which had been carefully hand-crafted to match the existing manual systems. No-one really believed that any database, however flexible, would exactly reproduce the programs on the BBC. There was obviously going to be a compromise between the economics of programming and 'acceptability' to the user. Exactly where this compromise should lie turned out to be a critical factor in the final choice of database system.

Which Database?

From a short-list of five, the choice narrowed down to two main contenders:
— FMS-80. Friendly, easy to use, helpful. This would do the 'simple' things quickly and easily, and had the great benefit that untrained staff could generate their own reports through the built-in menus.
— DBASE-II. The industry standard. Much more the professional programmer's tool; flexible and powerful, but definitely command-orientated with only the bare minimum of on-line help.

The winner was FMS-80; in the consultant's view the school system was so close to the classic 'personnel records' application that the 'so simple even the cat could use it' approach was adequate. If you did need to go off into special-purpose programming, there was always the 'extended language' facility (sort of Pascal-like) and the ″500 programming course to learn it. It was felt at the time that it would not be necessary to use this 'extended language' at all.

DBASE-II lost out on the grounds of being insufficiently 'state of the art' in terms of ease of use; in particular it was not thought to be sufficiently simple for untrained staff to use as a report-generator. This requirement had been heavily stressed in the consultant's brief; in the light of subsequent developments it is interesting to note how this one over-riding factor had influenced the choice, as it happened in the wrong direction!

Developing the Database System — Take 1

The School now had rather a stroke of luck; an ex-master had for some years run his own small computer outfit (accounting systems on PETs — that sort of thing). He had recently been 'bought out' by a larger firm, and was on the look-out for a change of job. After some heart-searching he agreed to act as systems analyst/programmer for the project. Of course he was less than convinced about the idea of database — he had after all been writing this sort of thing in machine code for the last 10 years.

As he started putting together the first file and screen layouts he quickly began to realise just how easy FMS-80 was making life; it really did make light work of things like simple screen editors and default reports! However as fast as he discovered the joys of database, he began to fall over the drawbacks! An alarming number of apparently straightforward jobs (those that the consultant had blithely assumed were 'standard') turned out to have just enough quirky bits that that dreaded 'extended language' was needed after all! Unfortunately the resulting programs were significantly harder to write (and follow) than they would have been in Basic!

At this same time it became clear that there were really very few genuinely 'ad hoc' reports at all; half a dozen 'selectable' lists were quite adequate to cover virtually everyone's requirements. The consultant's choice of FMS-80 began to look less and less viable, and the School was faced with a hard decision. Should they persist with a choice which seemed in retrospect to be wrong; should they throw away a considerable investment in software and training and begin again from scratch with DBASE-II; or should they scrap the idea of database altogether and revert to Basic?

Developing the Database System - Take 2

Of course this didn't by any means go back to square 1. The hard bits (the file design, the report specifications) had already been done, and they carried over unchanged from the previous development. As will be clear from the examples, it is by no means straightforward to program even the simplest of screen-based editors in DBASE-II; on the other hand it is quite possible to add 'one-off' calculations, for example to cross-check 'expected year of entry' and 'date of birth'.

```
IF global = 'S'
  STORE T TO notnext
  DO WHILE notnext
   @ 16,2 SAY ron+' Please make up to 3 entries for mode of selection, ';
     +'from the list above. '
   @ 17,2 SAY ' If you select fields 7,8 or 9 you may enter an inclusive ';
     +'range.         '+rof
   @ 18,2 SAY 'Please select your (up to) 3 fields now ' GET fld1 PICTURE '99'
   @ 18,50 GET fld2 PICTURE '99'
   @ 18,60 GET fld3 PICTURE '99'
   READ
   CLEAR GETS
   @ 16,1
   @ 17,1
   @ 18,1
   STORE VAL(fld1) TO v1
   STORE VAL(fld2) TO v2
   STORE VAL(fld3) TO v3
   IF v1‡11 .AND. v2 ‡11 .AND. v3‡11
     STORE F TO notnext
   ENDIF
  ENDDO notnext
  IF fld1 £ ' '
     STORE TRIM($(txts,(v1*9)-8,9)) TO tx1
     STORE TRIM($(flds,(v1*9)-8,9)) TO fn1
  ENDIF
  IF fld2 £ ' '
     STORE TRIM($(txts,(v2*9)-8,9)) TO tx2
     STORE TRIM($(flds,(v2*9)-8,9)) TO fn2
  ENDIF
  IF fld3 £ ' '
     STORE TRIM($(txts,(v3*9)-8,9)) TO tx3
     STORE TRIM($(flds,(v3*9)-8,9)) TO fn3
  ENDIF
  IF fld1 = ' ' .AND. fld2 = ' ' .AND. fld3 = ' '
     STORE 'G' TO global
  ENDIF
ENDIF global
```

Much the same applies to the reports, although a surprising number of 'internal' lists could be generated directly by DBASE-II. In fact as the system 'bedded in' the users began to realise that

```
    FOR ENTRY=85
    LIST REGNO,NAME,BIRTHDATE,FORM
```

...wasn't so hard after all, and was a good deal quicker than getting the same thing through the menus! So much for user-friendliness!!

As the development approached completion, it became increasingly obvious that DBASE-II really was giving considerable benefits over a 'do it yourself' approach. Oddly enough, the one unfulfilled expectation was for increased speed; many of the more complex tasks ran faster on the dear old BBC! This apart, the new systems measured up very well against that initial list of requirements:

- reliability. Surprisingly enough, even an old established package like DBASE-II has the odd bug! The indexing has the occasional inexplicable happening, and the supplied program editor (in practice you would probably use Wordstar anyway) sometimes crashes. However it is certainly beyond doubt that it would not be humanly possible to write such a sophisticated file system in Basic, and get it anything like as clean, within a year.
- speed of development. Most of the complex reports (e.g. an 'age pyramid' for the entire School) were effectively hand-crafted. The DBASE-II language proved itself very capable of the task, but saved little or no time over the same thing done in Basic. On the other hand the 'update and validate' routines and the simpler reports could take full advantage of the available macros, and probably took less than a tenth the time of a Basic equivalent.
- maintainability. This is an area where the real benefits have yet to appear. Of course there have already been several significant changes to the file structures, and the ability to re-load an existing database into a new structure (with fields extended or truncated as required) has been a major boon. Also, everyone agrees that the DBASE-II code is far easier to follow than any Basic equivalent.
- standard reports. As it turns out, many of these are simple 'subset' lists sorted in particular ways. DBASE-II can do this kind of thing with its eyes closed, and very quickly too. The non-standard ones are so non-standard that they virtually have to be programmed 'from the ground up' anyway; at least DBASE-II doesn't get in the way!
- speed. Once everything is set up correctly (of course you learn the wheezes as you go along) record selection and updating is extremely fast. There is a price to pay; a complete re-index of all 800 boys takes around half an hour to run. Fortunately this is a very rare occurrence! For the hand-crafted programs the situation is very different; because DBASE-II is an interpreter, and requires significantly more code than Basic to achieve the same job, it runs rather slower. Typically the old BBC programs beat the Apricot for speed on most of these 'special cases'.

In general there is no doubt that the 'Take 2' development has been extremely successful. In fact many fewer compromises have been made than were expected! It is interesting to note that many of the most important 'plus' points of DBASE-II:

- the powerful 'general-purpose' programming language
- the almost complete freedom from bugs
- the huge 'user-base', and hence the ready source of advice, literature, hints and tips, etc.

...were hardly considered in the original report. By contrast the main 'plus' points of FMS-80:

- the helpful front-end
- the 'menu-generated' reports

...were the basis of the original 'brief' on which it was chosen, and turned out pretty well irrelevant when the final system emerged!

Summary

Yes, database really does pay dividends; yes it really does pay to get the right database. No, you can't expect to get it right first time. Even if you take the sensible precaution of retaining an independent expert, he will probably ask all the wrong questions!

The Bedford School experience is particularly valuable, in that they got it wrong; recognised the fact; and had the courage to go back and start again. As a result they have developed a remarkably professional system extremely quickly. DBASE-II has been around a long time; for cheap and cheerful systems which need a modicum of hand-crafting it still looks head and shoulders above the competition.

APL◊385

* the number to phone for practical APL utilities.
* a compatible application-builder for mainframe or PC.
* all functions supplied with fully commented listings and supporting documentation.
* code and manuals by Adrian Smith, and that means readable, practical, and fun to use.

* *FSM◊385*	Simple, powerful and effective full-screen for mainframe & PC.
* *DRAW◊385*	The screen-design to go with it.
* *DB◊385*	A relational workspace. Nested arrays the cheap way!
* *GEN◊385*	Most of the best utilities from 'Design Handbook', and more.

Package for APL*PLUS/PC .... £50.00 each
(disk, manual and one user-guide)

Package for VS APL .... £125.00 each
(disk, manual and ten user-guides)

User-guide only .... £40.00 for 10

For more details please contact:

APL◊385, Brook House, Gilling East, York.
Telephone ... Ampleforth (04393) 385

In every industry, there has to be a leader

Personal APL – Our version of APL for the **Sinclair QL** brings full-powered APL at the lowest possible price.

PC-APL – MicroAPL supplies STSC's **APL*Plus/PC** with full backup, support, and ancillary software.

Departmental APL – For the ultimate in **performance and power,** our multi-user APL supermicros can handle even the largest applications.

Company-wide APL – With unrivalled experience in networking and communications, MicroAPL can offer a **comprehensive APL facility** linking PCs, supermicros and mainframes.

Consultancy in APL – MicroAPL's commitment to quality doesn't stop at our hardware products. Our uniquely experienced team of APL consultants will undertake any APL software project, from one day's help with a specific problem, to major team developments.

MICRO
APL

MicroAPL Limited

Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London SW8 5BP.
Telephone: 01-622 0395

TECHNICAL SECTION

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

TECHNICAL EDITORIAL

by Jonathan Barman and David Ziemann

What is it about programming style that rouses such strong emotions in people? APL symbols can cause apoplexy amongst those who are not used to them, and APL afficionados get very excited about code that is, to them, poorly written. This emotion is understandable if you take the view that hard to read code is by definition poorly written. "I'm great at writing code, but I can't make head or tail of this load of junk" is the first reaction, rather than "This looks clever, let's see if I can understand it". For instance, long functions with many globals usually, and we think rightly, elicit the first response. Short functions with no globals should warrant closer inspection before being classed as good, bad or indifferent. Of course, code which is very easy to understand may not necessarily be good APL; for example a function that loops on every row of a matrix rather than using the power of APL to process the whole matrix at once.

We recently heard of an APL programmer who was having some trouble making modifications to a system where there were two notable functions with over six and seven HUNDRED lines. We once had to maintain a workspace where the key functions had been named after the entire Arsenal football team! Actually it is often the apparently more sensibly named functions that cause problems. For example, 'BOX' has been used both for a function that converts a vector to a matrix, and for one that puts lines around a matrix for display. If the wrong function were copied into the workspace (from a utility library) it would cause a problem, in this case a small one; the functions are so different that the error would soon be detected. What happens though if the difference is small, and the error is not revealed during testing? A good example is the definition of a function often called 'ON' - how many similar, but different definitions have you seen?.

The concept of "programming style" can be thought of as applying to at least three levels of an APL system; the line level, function level and the workspace level. The line level refers to the style at the level of individual function lines, or fragments thereof, the function level to the contents of user defined functions and the workspace level to the interaction of all the components of the workspace.

Some examples may clarify this idea. At the line level multiple assignments are normally to be avoided because of the difficulty of debugging the code and because the line may not be restartable if an error occurs in the middle of the line. At the function level a good programmer will generally produce well named short functions that do not generate side effects. At the workspace level programming style is concerned with the relationships between functions. There's enough in this last topic for a whole book, and in fact Glenford J Myers has written one - although not with APL in mind. "Reliable Software Through Composite Design" is extremely useful in the design of APL programs, and formalises many of the ideas which we often just have a 'gut feel' about.

A further level where programming style could be said to operate is in the area of file design and access, again a large and involved subject, worthy of several books, let alone articles in VECTOR.

We would like VECTOR to become a forum for you to air your views on good and bad coding style. Let us know what you think, and if it generates some emotion, then that should make VECTOR more exciting reading!

VECTOR Publication Standards for APL Code

In order to make the production of VECTOR easier, the following rules should be adopted when preparing letters, articles and other contributions for publication.

Please do not include APL symbols within the text. If an APL symbol needs to be referred to, use its name; most APL textbooks have a list of symbols and their standard names.

Where variable or function names are referred to within the text they will be set in ordinary capitals. This means that sentences have to be carefully worded so that a function or variable name can be easily distinguished from any other words that need to appear in capitals.

APL statements and function listings will be reproduced from the submitted document whenever possible, and will be pasted onto a separate line between the text. This means that the listings should be of high quality; a 'daisy-wheel' printer with a carbon ribbon should be used. If you do not have a quality printer, we are prepared to re-set the APL ourselves if it is not too long, but we would much rather not do this as it is all too easy to make mistakes. Ideally we need two documents; the "readable" copy which shows what the final appearance is likely to be, and the "printer's" copy which we actually send to the printers. The "printer's" copy needs to have all the APL completely separated from the text. The easiest method is to leave three blank lines in the text where the APL is to appear, and to have a separate sheet containing all the APL; each section also separated by three lines. A simple numbering scheme can then be used to show where the APL block is to appear in the text. These APL characters will be photo-reduced and inserted into the text body.

Drawings, figures and black and white photographs are welcome, as they help to break up the text and make articles more visually interesting. Please prepare each figure on a separate sheet of paper, and give it a number and caption. The caption will be typeset and will appear at the top of each figure, which will be presented in a box as near as possible to the reference in the text. Do not worry about the size of the drawing, as the printers will photo-reduce it to fit the size of the page or the space available. Your text should refer to the drawings strictly by sequential figure numbers.

Footnotes and references will be grouped together at the end of the article, so please avoid writing text that requires a footnote to be on a particular page.

Introduction to Contributed Articles

As APL2 inevitably becomes more popular the need for us to stop procrastinating and get to grips with it increases. To help us do this Norman Thomson of IBM Hursley Park has written our first contributed article in this issue. "A Guide to APL2 Nested Arrays" is a tutorial which takes us to quite a sophisticated level of expertise via a number of "five-finger exercises" — the APL2 equivalent of piano practice.

Following the nested arrays theme, John Scholes from Dyadic Systems presents a short excursion into some of the second generation features of their APL. "Operators and Nested Arrays in Dyalog APL" nicely demonstrates the power of these extensions.

Per Hultin and John Hagger address the issue of choosing the right horses for courses and explain why they have taken the approach of developing assembler routines to improve APL performance by speeding up many of those important utilities.

Technical Correspondence

From Adrian Smith 15 May 1985

Sir: I am not entirely convinced by your first example in 'Surely there must be a better way' from Vector 1.3. I agree that the original function is rather awful, but in some ways your suggested numeric versions obscure the structure of the problem. They could also make 'trivial' changes unnecessarily hard, as the whole algorithm would need re-thinking. Maybe something like:

```
∇ R←CHECK X
 ⍝ RETURN APPROPRIATE FLAG DEPENDING ON VALUES IN X[1 2]
  R←0
  →(Comb,Ster,Doll,0)IF 0 1 2 =1↑X
 Comb:R← 5 6 [¯1↑X] ⋄ →0
 Ster:R← 1 2 [¯1↑X] ⋄ →0
 Doll:R← 5 6 [¯1↑X] ⋄ →0
∇
```

. . . note the multi-way IF, which I find an absolutely invaluable extension when faced with this sort of problem.

Yours sincerely,

Adrian Smith
Operational Research,
Rowntree Mackintosh,
YORK YO1 1XY.

Editor: Absolutely — I'm surprised that no-one else complained! It could be that I over-reacted in boiling down the original 22-line function to:

```
R←5 6 1 2 5 6[2⊥C]
```

The point is that the author had failed to spot a pattern underlying the problem and as a result produced uncommented and verbose code. Of course, the most important thing is that the APL should not be obscure, but understandable by another programmer (who might have to maintain it). This can be achieved both by writing clear code and by using comments as well, which your function does. By the way, I agree that the multi-way IF function is a very useful tool — and an underused one.

From Andrew Wiggins 17 April 1985

Sir: I enclose a problem that I have had around for many months. It can easily be solved using a loop, but I have yet to find a way without the use of looping. I would therefore like to hand it over to the other readers of VECTOR, so that they may attempt it.

Yours faithfully,

Andrew Wiggins
Lombard North Central PLC,
Lombard House,
London W1A 1EU.

Editor: Andrew's problem is presented in this issue's "Surely there must be a better way" column.

From Mark Bassett 13 May 1985

Sir: Here is a suggestion for a competition that you might like to include in a future issue of VECTOR. The problem had its origin in the need to display lists of names, grouped under various headings, in a way that was a little less dazzling to the eye than a simple columnar layout.

This led to the invention of a function, WORDWRAP, that would reshape text into a matrix of specified width with each row left-justified and no word being 'broken' at the right-hand margin unless it was too long to fit on one line.

The attached printout shows two examples of its use, taken from Lewis Carrol, with a suggested implementation below (the best I could find). The idea used was to build up the result row by row: take characters from the input text N at a time, remove any trailing non-blanks and catenate onto the output, unless this makes the next character a blank in which case we've reached the end of a word and should catenate without removing anything.

It is the need to worry about the beginning of the next line while constructing the current one that results in the function scanning the input in blocks of size N + 1.

No amount of search sufficed to discover a non-looping method so this is one obvious improvement entrants could make; if looping really is intrinsic to the problem then perhaps less work could be done inside the loop.

Yours faithfully,

M.S. Bassett
20 Coval Lane,
Chelmsford,
Essex,
CM1 1TD.

Editor: It seemed more appropriate to include Mark's problem in the "Surely there must be a better way" column, and so that's where it is, along with some examples and Mark's solution.

SURELY THERE MUST BE A BETTER WAY

compiled by David Ziemann

This issue we have two STMBABWs; one from Andrew Wiggins of Lombard North Central, and one sent in by Mark Bassett. In their letters, which appear on the Technical Correspondence page, they both explain that they have found looping solutions to their respective problems, but have not been able to find non-looping ones. Here's Andrew to kick off with his problem:

"The problem is accountancy based, but don't let that put you off. It involves two pieces of information:

1) Income
2) Depreciation

The rules are that:

a) Depreciation must never exceed income, (although they may be equal).
b) Any excess of depreciation over income must be carried forward and used as soon as possible AFTER the period(s) in which any excess may occur.

Here is an example problem:

Year	1	2	3	4	5	6	7	8	9	10
Income	100	110	120	130	140	150	160	170	180	190
Depn.	70	130	140	130	120	60	180	170	150	230
Profit/(Loss)	30	(20)	(20)	0	20	90	(20)	0	30	(40)

. . . and its required solution:

Year	1	2	3	4	5	6	7	8	9	10
Income	100	110	120	130	140	150	160	170	180	190
Depn.	70	110	120	130	140	80	160	170	170	190
Profit/(Loss)	30	0	0	0	0	70	0	0	10	0
Carried forward	0	20	40	40	20	0	20	20	0	40

Presumably Andrew will be happy with a solution that yields the Profit(Loss) and Carried forward rows from the required solution table shown above.

Mark Bassett wants a function <WORDWRAP> that reshapes text into a left justified matrix of specified width with no uneccessary word breaks. He provides the following examples, along with his version of the function:

```
      X←' Twas brillig and the slithy toves '
      ,X←X,'did gyre and gimble in the wabe. '
 Twas brillig and the slithy toves did gyre and gimble in the wabe.
      Y←'All mimsy were the borogroves '
      ,Y←Y,'and the momerathes outgrabe.'
All mimsy were the borogroves and the momerathes outgrabe.
      20 WORDWRAP X
Twas brillig and the
slithy toves did
gyre and gimble in
the wabe.
      8 WORDWRAP Y
All
mimsy
were the
borogrov
es and
the
momerath
es
outgrabe
.
```

```
     ∇ R←N WORDWRAP A;C;D;V;X
       ⍝ Wrap text <A> into a matrix <R> of width <N>
       ⍝ If at all possible, words are not broken.
        D←1↑0⍴V←,A
        R←(0,N)⍴''
      L1:→(0=⍴V)/0
        V←(∨\V≠D)/V
        X←(N+1)↑V
        C←N+1-+/∧\⌽X≠D
        C←C-N<C←C+N×C=0
        R←R,[1]N↑C↑X
        V←C↓V
        →L1
     ∇
```

Can anyone improve on this, in terms of either algorithmic elegance or execution speed? Let us know how you get on.

PRIZE COMPETITION RESULT: TEST YOUR SKILL

by David Ziemann

To recap; in our mythical consultancy we have a consultants' skills matrix as follows:

```
      CONS
1 2 3 7 0 0
1 3 7 9 2 6
7 9 0 0 0 0
1 6 5 3 9 0
```

where each row represents one consultant's skills, and each non-zero entry is an index into a table of skill descriptions. For example, the third consultant(row) has skills 7 and 9. Notice that zeros are used to pad the shorter rows. Also, we have a matrix of skills required for the jobs that have arisen, as follows:

```
      JOBS
1 2 3
7 9 0
1 4 0
1 3 0
1 6 3
```

The fifth job (last row) requires skills 1, 6 and 3 for example, and it could in fact be performed by the fourth consultant, who has the relevant skills. The problem was to write a function to generate a boolean matrix that specifies which jobs can be tackled by each consultant, i.e.:

```
      JOBS SKILLSMATCH CONS
1 1 0 0
0 1 1 0
0 0 0 0
1 1 0 1
0 1 0 1
```

Although the consultancy has a potentially large number of customers and employees, the number of unique skills is low, and it was stated that contestants would not be penalised for offering solutions that assumed no more than 10 skills. Workspace demands, execution speed, robustness and intelligibility were named as the factors that would determine the winners.

Over thirty functions arrived before the closing date, with entries from many parts of the planet. The entries split into two broad types; those that are 'pure' APL solutions and work for any number of skills, and those that take the hint about 10 skills being acceptable. The majority of functions entered fall into the first group, so let's deal with them first. The 'do it all at once' approach was used by many entrants; Henri Schueler from Toronto, and Bengt Lingren from Sweden for example, produced the following solution, as did four others:

```
      ∇ R←J SM1 C
  [1]  ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
  [2]   R←∧/[2]∨/J∘.=C,0
      ∇
```

A column of zeros is catenated to the consultants' skills table to ensure that everyone has skill zero, which is used for padding in the jobs matrix. The OR reduction of the outer product shows whether a consultant has each of the required skills, and the AND reduction tells us if the consultant has ALL the required skills.

Simon Barker from London came up with this noteworthy algorithm:

```
      ∇ R←A SM2 B
  [1]  ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <A> AND CONSULTANTS <B>
  [2]   R←⍉∧/∨⌿ 3 4 2 1 ⍉A∘.=B,0
      ∇
```

The following origin-independent answer was submitted by David Quas from Bristol:

```
      ∇ RES←A SM3 B
  [1]  ⍝ <RES> IS SKILLS MATCH MATRIX FOR JOBS <A> AND CONSULTANTS <B>
  [2]   RES←(¯1↑⍴A)=+/[1+⎕IO]∨/A∘.=B,0
      ∇
```

These concise and elegant algorithms work for any number of skills, but can be very greedy on workspace for large arguments - after all, an outer product followed by a couple of reductions invariable means that we are throwing a lot of generated data away. This waste can be avoided by coding a looping solution, where the correctly shaped result matrix is initially established and its contents modified within the loop. This approach was taken by David Quas who provided us with this function:

```
      ∇ RES←A SM4 B;⎕IO;N;MAX
  [1]  ⍝ <RES> IS SKILLS MATCH MATRIX FOR JOBS <A> AND CONSULTANTS <B>
  [2]   RES←((1↑⍴A),1↑⍴B)⍴0
  [3]   ⎕IO←1
  [4]   B←B,0
  [5]   N←0
  [6]   MAX←1↑⍴B
  [7]  LOOP:→(MAX<N←N+1)/0
  [8]   RES[;N]←∧/A∊B[N;]
  [9]   →LOOP
      ∇
```

Notice that one iteration is needed for each row in the consultants' table, and that a leading loop test is performed to ensure the correct behaviour with empty arguments. This type of looping solution takes less workspace to run and can also be very much faster than the previous examples. In the words of Simon Barker:

"The fact is, the looping solution is much faster (and more space conserving) than the one-line non-looping solution. To me, the non-looping answer is the most pleasing because it is the purest of all APL solutions and its conciseness could not be achieved in a million years in any other language. Sadly, however, the thoroughness of comparison between elements of different arrays is its downfall, as a lot of what it does is not necessary."

Some people decided to make use of the fact that the required solution needed to work with no more than 10 skills. Curtis Jones from San Jose, USA and Phil Last from London discovered that a substantial speed-up over the one-liner could be achieved by coding

```
      ∇ Z←J SM5 C
  [1]  ⍝ <Z> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
  [2]   J←∨/[2]J∘.=⍳10
  [3]   C←∨/[2]C∘.=⍳10
  [4]   Z←J∧.≤⍉C
      ∇
```

Similar techniques were used by Henri Brudzewsky from Denmark and Ken Goralski from London, while Richard Fisher replaced line 4 by

```
Z←~J∧.∨⍉~C
```

which runs faster on some APLs. A conceptual leap was made by Greg Mateja from Hartford, USA who provided this:

```
      ∇ R←J SM6 C;N
 [1]   ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
 [2]    C←C×C≤N←1+⌈/,J
 [3]    J←⍉ ¯1 0 ↓(N⍴2)⊤+/(J≠0)×2*J
 [4]    C← ¯1 0 ↓(N⍴2)⊤+/(C≠0)×2*C
 [5]    R←J∧.≤C
      ∇
```

Greg receives a special commendation for supplying a 52-line function, all but 4 lines of which were comments! The technique he uses involves raising 2 to the power of all the non-zero elements in each skills matrix, and summing. Converting the result back to base 2 then provides a boolean with every possible skill flagged either 1 or 0. (This works because there is no significance to a skill appearing more than once in a row). These matrices are then directly compared by using the appropriate inner product. Note that the method fails when the jobs skills numbers get too large. Mike Day from London used a similar approach, but discovered that in MIPS APL the code fragment

```
(1-W)∧.∨Y
```

runs more quickly than the more obvious

```
W∧.≤Y
```

His function is as follows:

```
     ∇ R←JOBS SM7 CONS;MAXSKILL;TWOS;TWOPOWERS;⎕IO
 [1]  ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <JOBS> AND CONSULTANTS <CONS>
 [2]   TWOPOWERS← 0 1 ,×\TWOS←(MAXSKILL←⌈/(,JOBS),(,CONS),⎕IO←0)⍴2
 [3]   R←(1-⍉TWOS⊤+/TWOPOWERS[JOBS])∧.∨TWOS⊤+/TWOPOWERS[CONS]
     ∇
```

Mike also presented another 'mathematically attractive' algorithm, which he included for comparison; instead of powers of 2, a base 3 representation of the sums of powers of 3 is used. If any 2s exist in the base 3 representation then there must be a mis-match between the required and actual skills. In fact, this will work for any base higher than 2, but larger numbers would soon cause problems with precision. Again, time and memory requirements are excessive. For interest, here is such a function:

```
      ∇ R←J SM8 C;M;⎕IO
 [1]   ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
 [2]    M←1+⌈/(,J),(,C),⎕IO←0
 [3]    R←⌈0,3*⍳M
 [4]    R←2∧.>(M⍴3)⊤(R[M]-+/R[J])∘.++/R[C]
      ∇
```

Well, we're almost home and dry, but first, another looping solution. This one was sent in by Adrian Smith, who accepts that he is ineligible to enter, being on the VECTOR working group! It's interesting because it loops not on the rows of the jobs matrix, but on the number of skills in it. Here it is:

```
     ∇ MAT←JB SM9 CN;lab;CT;SK
       ⍝ <MAT> IS SKILLS MATCH MATRIX FOR JOBS <JB> AND CONSULTANTS <CN>
        SK←⌈/,JB
        MAT←((1↑⍴JB),1↑⍴CN)⍴1
       Loop:→lab←1+(SK⍴Loop),End,CT←1
        MAT←MAT∧(JB∨.=CT)∘.⍲(CN∧.≠CT)
       End:→lab[CT←CT+1]
     ∇
```

A modification would need to be made for correct operation with empty arguments. Adrian would like to know if anyone else has a sufficiently warped brain to have tackled it this way (Frankly I doubt it...). He says:

"It took from Potters Bar to just north of Peterborough to dream this up, and most of the way to Doncaster to check it! As long as you meant that hint about small numbers of skills, you will find that it goes quite quickly. It would be fascinating to see some timings against one of the jot.epsilon solutions from APL2!!"

The idea of using the skills matrices as indices into a set of numbers, as seen above, can be pursued further. What happens if instead of indexing from powers of 2 or 3, we use prime numbers? By taking the product over such a set we create a unique number; one that could only have been generated from the original set. Let's look at a simple example. Can the consultant with skills 1, 3, 7, 9, 2 and 6 attempt the job for which skills 1, 6 and 3 are required? The products over the set of primes are as follows:

```
      P←1 2 3 5 7 11 13 17 19
1 2 3 5 7 11 13 17 19
      ×/P[1 3 7 9 2 6]
16302
      ×/P[1 6 3]
33
```

Because these results are unique, if the first divides exactly by the second, then its original set is a superset of the second's. And so the answer to the above question is "YES" because

```
      33|16302
0
```

Here is the complete function:

```
     ∇ Z←J SM10 C;P;⎕IO
       ⍝ <Z> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
        ⎕IO←0
        P← 1 2 3 5 7 11 13 17 19 23
        Z←0=(×/P[J])∘.|×/P[C]
     ∇
```

This trick is often colloquially referred to as 'Goedelisation', after the mathematician Goedel. The use of index origin 0 in the function provides a neat way of correctly ignoring the effect of the zeros used for padding — the contribution to the product being 1. You have to be sure to use enough primes for the index expressions to succeed; in origin-0 this is one plus the highest skill number. The algorithm is elegant, concise and economic; the amount of space required is dependent on the size of the result, not on some intermediate structure. It is also the fastest, as can be seen by the sample timings below.

Three contestants supplied this function; they are Richard Fisher from Epping, Australia, Derek Wilson from York and Bernd Fohlmeister from Cologne, Germany. Phil Last's entry was very similar and Greg Mateja provided a close variant. Deciding how to split the booty is a soul-searching torment, but the postmark dates decide, and so £30 goes to Richard and £10 each to Derek and Bernd.

The following timings were made on version 4.1 of APL*PLUS/PC on an IBM PC with the 8087 enabled. A JOBS matrix of shape 20x5 and a CONS matrix of shape 40x6 were used, to yield a result shape of 20x40. All times are in seconds:

SM10	2.3
SM4	4.0
SM9	4.4
SM7	4.6
SM5	5.3
SM6	6.1
SM1	13.4
SM3	13.4
SM2	16.6
SM8	22.4

This ordering of functions according to speed may vary when other APL interpreters are used, but <SM10> will probably remain unbeaten.

Two respondents provided solutions written in non-standard APL. This Sharp APL answer was sent in by someone whose surname is Baronet (first name unfortunately unreadable), from Toronto, Canada:

```
      ∇ R←J SMIPSA C
  [1]  ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
  [2]   R←⍉∧/J∊⍤ 2 1 C,0
      ∇
```

The right argument of the 'ON' operator is the integer vector 2 1 which controls the way the membership function is applied; the 1 means that the function is applied to vectors of the matrix right argument. The 2 causes the left argument, J , to be treated as a matrix (which in this case it is anyway). Apparently the function runs about 40% faster than <SM1>under Sharp APL, and clearly takes less room.
Norman Thomson (whose tutorial on APL2 nested arrays appears later in this issue) suggests the following APL2 function:

```
      ∇ R←J SMAPL2 C
  [1]  ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
  [2]   R←∧/¨((⊂[2]J)~¨0)∘.∊⊂[2]C
      ∇
```

The tilde symbol represents the function 'without'. Norman also thoughtfully provides us with this translation into something he calls APL1:

```
      ∇ R←J SMAPL1 C;X
  [1]  ⍝ <R> IS SKILLS MATCH MATRIX FOR JOBS <J> AND CONSULTANTS <C>
  [2]   R← 1 1 2 ⍉(+/X)∘.=(X←⍉∨/(⍳10)∘.=J)+.∧∨/(⍳10)∘.=C
      ∇
```

although he says he's sure that other VECTOR readers could do better.
What a shame we don't have an interpreter that can run all flavours of APL - we could come up with some interesting comparative timings!

Please note that the author's original comment lines have not been included in the functions as presented here.

By the way, the above functions show a great deal of creativity in the area of the inner product operator, with all these products appearing at least once:

```
∨.∧
∧.≤
∧.∨
∧.>
∨.=
∧.≠
+.∧
```

Also worthy of note are following boolean identities, which the competition has revealed:

```
A∧.≤B    ←→    ~A∨.∧~B    ←→    (1-A)∧.∨B
```

Studying how they work can help us to better understand APL and to use it more effectively, and is well worth the effort involved.

PRIZE COMPETITION: WRAP UP

by David Ziemann

Very often APL programmers will represent rows in a notional text matrix as the concatenation of vectors delimited by carriage return characters. This representation usually takes up less workspace because trailing blanks are not included.

For example, many APLs use a system function quadVR to return the visual representation of a user-defined function. This is a character vector containing imbedded carriage returns that displays as a neat function listing. Unfortunately, such displays often exceed some arbitrary width, because lines between the carriage returns are too long. In such cases it is desirable to wrap the offending lines by inserting appropriate extra carriage return characters. Example output could be:

```
      ,S←'IN PARADISE',cr,'HERONS',cr,'APPROACH FROM THE LEFT.'
IN PARADISE
HERONS
APPROACH FROM THE LEFT.

      10 0 WRAP cr,S
IN PARADIS
E
HERONS
APPROACH F
ROM THE LE
FT.
```

In this example the result is similar to the argument except that three extra carriage returns have been inserted; one after the 'S' of 'PARADISE', one after the 'F' of 'FROM' and one after the 'E' of 'LEFT'. Notice that the function <WRAP> is quite general, with the first element of its right argument providing the delimiter character to be used. The left argument specifies the maximum allowable length substring between delimiters in the result, followed by the amount of indentation required. The indentation amount determines the number of leading blanks in each wrapped segment. For example:

```
      11 2 WRAP cr,S
IN PARADISE
HERONS
APPROACH FR
  OM THE LE
  FT.
```

The problem is to write such a function <WRAP>. As usual, the winning programs must be ISO APL standard conforming (use no extensions like replicate, nested arrays or special quad functions, etc.). Each entrant must submit only one competition entry, but may include alternatives for comparison and possible publication. These alternatives may be written in ANY APL, so let's see some APL2 (etc) solutions too!

Entries will be judged on robustness, resource requirements, generality and intelligibility. A first prize of £30 and two others of £10 each will be sent to the winners.

The closing date for entries is 25th October 1985.

MetaPack... Everything you need for APL on the PC.

MetaPack is the utilities package that has been designed especially for users of APL★PLUS/PC.

MetaPack comprises a number of programs, each designed to help you build better systems in less time.

MetaPack will increase your productivity as an APL systems builder, and enhance the function and appearance of your systems.

The **MetaPack** programs are:

MetaScreen	— the popular AP124 based full-screen management system. The speed-critical modules have been written in assembler, making **MetaScreen** ideal for running mainframe applications on the PC.
NI	— for full-screen input and editing of APL numeric variables.
Browse	— an attractive and sophisticated method of viewing reports or text files on the screen by horizontal or vertical scrolling.
ToolBox	— an invaluable collection of APL utilities, including table-lookup and vector-matrix conversion functions written in assembler.
MetaWS	— the program filing system for easy management and documentation of large APL systems.

MetaPack programs have been written in a combination of APL, C and Assembler. This means that your systems can benefit from **MetaPack's** advanced features yet still run with realistic execution times.

The programs may be used either as stand-alone utilities or as an integrated part of your systems.

Whether you're a professional APL systems builder or an end user who has discovered the power of APL, you will find **MetaPack** an indispensible aid to building better systems.

MetaPack — £495 **MetaScreen** (available separately) — **£245**
For IBM PC XT AT, COMPAQ & Compatibles

APL★PLUS/PC is a registered trademark of **STSC, Inc.**

MTS **MetaTechnics Systems Ltd**
Unit 216V, 62 Tritton Road, London SE21 8DE, 01-670 7959
The Systems Builders

Competition Rules

- Entries must be in legible English or APL as appropriate and should preferably be machine produced.
- Entrants must declare the type of computer and the version and release level of the APL interpreter on which their functions were written.
- The date and your full name and address should appear on each sheet of your entry.
- Entries should be physically separate from other contributions such as letters, and should be clearly marked 'Competition Entry'
- All submissions should be sent to the editor.
- Those on the committee, activities group or journal group of the British APL Association are ineligible.
- DOS format diskettes containing APL*PLUS, IBM or Sharp APL workspaces are acceptable, Diskettes will be returned.
- Unless otherwise stated, you should submit only one entry. We encourage submission of alternative approaches, but you must indicate clearly which answer is the competition entry.

A GUIDE TO APL2 NESTED ARRAYS

by Norman Thomson

The 'brave new world' of second generation APLs (such as IBM's APL2 on which this article is based) is dominated by two ideas — nested arrays and operator extension. Nested arrays are in principle such a simple idea that only a few minutes are needed for a practised APL user to read and absorb the technical specifications of the enclose function, the disclose function and the each operator which together embody most of the essence of nested arrays. Yet the practice needed to acquire fluency in application defies all initial belief. How long does it take to become an effective and accomplished user of nested arrays? 1 year? 2 years? 5 years? The path of the typical APL user presented with a second generation APL is an initial spell of rapturous excitement over the potential power of the new language, followed by a period of increasing insecurity in which full mastery begins to recede like an unattainable goal.

Why does it take so long to feel comfortable with nested arrays? Partly — perhaps mainly — because having to think *in* depth *about* depth adds an additional dimension to programming, analogous to the mental leap needed in geometry to move from thinking in 2 dimensions to thinking in 3. To know the shape of an object in APL1 (hereafter used to refer to first generation APLs) is to know its all. In APL2 one has to know both shape and depth. (Rank, being shape of shape, is subsumed in shape.) So it makes sense to have in every APL2 workspace a function DR (Depth-Rho).

```
DR: (≡R),⍴R
```

which can be used as an interactive check on the structure of APL2 objects. Since depth is always an integer scalar the result is never ambiguous. If DR returns a single integer, then the shape vector is iota zero.

For example, define:

```
W←5 3⍴'ANDBOYCANDADEAT'

DR W   is  1 5 3    and

DR ⊂W  is  2
```

(or more accurately 2 followed by an invisible iota zero.)

W and its enclose are thus enormously different although when printed they look practically identical.

A major problem with using nested arrays is that a tiny difference in code can make a large difference in structure, and consequently an important step in acquiring accomplishment in handling nested arrays is the recognition of differences between similar expressions, some sequences of which will form the main basis of this article.

Learning APL2 is in my experience associated with the evolution of a personalised informal vocabulary supplementing the formal vocabulary of concepts. As an example, the "each" operator will be described as a tool for "loop-avoidance" on the one hand, and for "shell-penetration" on the other. Developing one's informal vocabulary is all part of the APL2 learning process!

Motivation for using nested arrays

The appeal of nested arrays is the periodic realisation that you can say what you really wanted to say instead of fumbling for contrivances around the limitations of APL1.

Such moments of revelation seem to arise mainly in two areas — mathematics and text. The application of nested arrays in the data-base context — e.g the realisation that a bibliography, say, or a personnel file is conceptually nothing other than a nested array — is I think sufficiently well-known to require no further exposition. An example of mathematical context is however worth giving.

Suppose that we have an excellent eigenanalysis algorithm such as is supplied by IBM in the MATHFNS workspace distributed with APL2, and that this returns an (N + 1)xN matrix whose first row is the N eigenvalues, and whose columns (excluding the first row element) are the N eigenvectors.

Thinking *mathematically,* if M is a square matrix

```
E←EIGEN M
```

is conceived as a 2xN matrix:

$eval_1$	$eval_2$	...	$eval_N$
$evec_1$	$evec_2$	...	$evec_N$

(Index origin 1 is assumed throughout)

It is natural to enclose *rows* so that columns retain their identity.

```
⊂[1]E
```

is thus a DR = 2,N structure of the eigen-pairs:

$$\boxed{eval_1 \quad evec_1} \quad \boxed{eval_2 \quad evec_2} \quad \cdots\cdots \quad \boxed{eval_N \quad evec_N}$$

```
⊂[1]1↓[1]E
```

or equivalentlv

```
1↓¨⊂[1]E
```

is the vector of N eigenvectors:

$$\boxed{evec_1} \quad \boxed{evec_2} \quad \cdots\cdots \quad \boxed{evec_N}$$

The second of the above expressions shows "each" in its "loop-avoidance" role.

Suppose on the other hand we wish to obtain the value of the determinant of M by multiplying together the eigenvalues, the natural way to think about this is first to enclose *columns,* so that rows retain their identity:

```
        ⊂[2]E
```

Of course only the first element in this (N + 1)-element vector is of any interest, and at first sight it might seem that

```
        ×/1↑⊂[2]E
```

defines the value of the determinant. But think — the arguments of "times reduce" is a one-element vector, and so its value remains unchanged as per APL1. What we have to do is to penetrate the shell of enclosure and "times reduce" the *contents* of the right argument:

```
        ×/¨1↑⊂[2]E
```

which is indeed the determinant idiom. Of course

```
        1↑×/E
```

has the same effect without any resource to APL2, however this involves, both conceptually and in fact, carrying out the meaningless multiplications of the 1st., 2nd, etc. eigenvector components. Arguably the APL2 expression gives a more natural expression to the underlying mathematical idea.

Notice how in the APL2 expression the "each" is not a way of "loop avoidance" (the loop is of size 1), but of "shell-penetration". I am not suggesting that the each operator has two different meanings — rather that the same underlying semantics have different significance in different contexts, in much the same way as the early users of APL1 discovered that in the right context "times iota" means "IF". It is worth spending a moment or two considering the common underlying semantics. The definition of "F-each" R is that its Ith element is

```
        ⊂F⊃ R[I]
```

— note the pleasing symmetry of the enclose and disclose symbols.

In the first example of "each" given above, F stands for "one drop". First the right argument is disclosed one element at a time, then F is applied to each element and finally all the elements are sewn up again with the enclose. The technique of opening up, applying F and then sewing together again is exactly the same in the second case with the F now representing "times reduce", but now there is only one element, and "each" is the mechanism for doing work inside the shell.

The effect of partial enclosure on shape is worth considering. Recall that

```
        ρE       is (N+1),N.

        ρ⊂[1]E   is  N  and each element has shape N+1;
        ρ⊂[2]E   is N+1 and each element has shape  N,
```

which demonstrates the rule that whatever the axis of enclosure, the corresponding element is deleted from the shape vector to obtain the shape of the result, and reappears as the shape of the enclosed elements.

More generally, and more informally whatever the axis *vector* of enclosure, the corresponding elements of the shape vector are deleted from the *external* shape, and conveyed to the *internal* structure, which is available incidentally as the "each" of ΔR.

Some practice in basics

At this stage it is appropriate to introduce some elementary exercises with enclose and "each".

Define

```
      V←4 5 6
```

What are the following:

```
(a)   2 3⍴¨V    ?

(b)   2 3⍴⊂V    ?

(c)   2 3⍴¨⊂V   ?

(d)   2 3⍴⊂¨V   ?
```

(A thought — why not quickly cover up the immediately following lines, so that you *think* the answers out before either reading them or tapping out on your APL2 system.)

(a) introduces a new addition to the "each" theme, namely a non-scalar left argument. To accommodate this the definition expands to give the Ith element of L (F-each) R as

```
⊂(⊃L[I]) F ⊃R[I]
```

that is, *both* left and right arguments are opened up, a loop through the element-by-element function applications takes place, and then every element is sewn up as before.

In the present instance there are 2 elements on the left, 3 on the right so the result is LENGTH ERROR.

In (b) and (c) the right argument is a scalar and so scalar expansion as per APL1 guarantees that no such problems arise. For (b) the result is

```
┌─────┐ ┌─────┐ ┌─────┐
│4 5 6│ │4 5 6│ │4 5 6│
└─────┘ └─────┘ └─────┘

┌─────┐ ┌─────┐ ┌─────┐
│4 5 6│ │4 5 6│ │4 5 6│
└─────┘ └─────┘ └─────┘
```

(c) means loop through two separate applications of "reshape":

```
┌───┐ ┌─────┐
│4 5│ │4 5 6│
└───┘ └─────┘
```

(Note at this point that the number of box boundaries that must be crossed in diagrams such as the above in order to get to the "core" is one less than the depth — if you use the DISPLAY function of APL2 there is a further outer box, so that the number of crossings is exactly equal to the depth. The final crossing corresponds to the contribution of 1 to depth arising from "non-scalarness".)

In (d) the effects of enclose and "each" cancel each other out, since enclose means "make-a-scalar-of" whereas each element of V is a scalar anyway. Hence this is just the same as "2 3 reshape V":

```
4 5 6
4 5 6
```

Now try

```
(e)   (⊂2 3)⍴V

(f)   (⊂2 3)⍴⊂V

(g)   (⊂2 3)⍴¨V

(h)   (⊂2 3)⍴¨⊂V
```

Note that "reshape" (as opposed to the *derived function* "reshape-each" must have a *simple* (i.e. non-nested) left argument, and so both (e) and (f) are DOMAIN ERRORS. The distinction between a function and the derived function obtained by application of an operator is crucial to understanding APL2 expressions.

In (g) the derived function has a scalar left argument and a vector right argument, so the former is scalar-expanded as in APL1 and element by element execution gives a 3-element vector:

```
┌─────┐ ┌─────┐ ┌─────┐
│4 4 4│ │5 5 5│ │6 6 6│
│4 4 4│ │5 5 5│ │6 6 6│
└─────┘ └─────┘ └─────┘
```

In (h) "reshape-each" has 2 scalar arguments — both are opened up for function application and then sewn together again to give

```
┌───────┐
│4  5  6│
│4  5  6│
└───────┘
```

that is, an object with DR = 2.

Partial enclosure and disclosure

Here is another set of simple exercises to clarify partial enclose and disclose. Suppose

```
M←2 3⍴⍳6
```

What are

```
(a)   V1←⌽¨⊂[1]M ?

(b)   V2←⌽¨⊂[2]M ?
```

(a) Rows lose their identity so the meaning is "rotate-each" a 3-element vector of columns:

```
V1:  ┌────┐ ┌────┐ ┌────┐    DR =  2 3
     │4  1│ │5  2│ │6  3│
     └────┘ └────┘ └────┘
```

(b) Right argument of "rotate-each" is a 2-element vector of columns and so we have

```
V2:  ┌───────┐  ┌───────┐    DR =  2  2
     │3  2  1│  │6  5  4│
     └───────┘  └───────┘
```

A short digression is appropriate here to consider the important function attributes: *depth-increasing* and *depth-decreasing*.

For example:

```
↑V1 is 4 1          DR =  1 2

↑¨V1 is 4 5 6       DR =  1 3
```

Note that the monadic form of "take" is called "first" and that both it and its derived function "first-each" are depth-decreasing.

```
Similarly      2⊃V1 is 5 2      DR = 1 2

               2 2⊃5 2 is 2     DR = 0 .
```

Observe that the dyadic form of disclose is called "pick", so that the first of these expressions is read "2 disclose V1". Then notice how pick is not merely a depth-decreasing function, but can be a multiply depth-decreasing function when its left argument, as in the second example, is a *path.*

Just as enclose is a depth-increasing function, so disclose is a depth-decreasing function. Disclosing along an axis (partial disclosure) means taking elements from the internal structure and placing them in the shape vector of the external structure — where to place them is determined by the axis specification. Thus

```
⊃[1]V1
```

takes the internal structure (2) and places it in the first position of the shape of the external structure, resulting in

```
⌽[1]M,
```

which demonstrates that enclose and disclose along the same axis are inverse functions.

```
⊃[2]V1
```

on the other hand places the 2 second in the shape vector of the external structure resulting in:

```
⍉⌽[1]M.
```

Similar considerations show that:

```
      ⊃[2]V2 ↔ ⌽[2]M
and   ⊃[1]V2 ↔ ⍉⌽[2]M.
```

To summarise — partial enclosure, function application, then disclosure along the *same* axis is equivalent to function application along that axis; if the disclosure is along a different axis, the result is equivalent to a transposition.

Practice with characters

The final set of exercises relate to a text example. The problem is to insert spaces into the matrix W defined at the start as

```
AND
BOY
CAN
DAD
EAT
```

A simple well-defined problem? — it may surprise you to know that there are at least 9 ways in which the little word "into" in the previous sentence can be interpreted. As before you might like to *think* through the exercises before reading the answers or keying in the questions. Writing down the DR of the answer is also recommended as a good discipline for getting the answer right first time!

```
(a)  1 0 1 0 1\W                     ?

(b)  1 0 1 0 1\⊂W                    ?

(c)  1 0 1 0 1\¨W                    ?

(d)  1 0 1 0 1\⊂[1]W                 ?

(e)  1 0 1 0 1 0 1 0 1\⊂[2]W         ?

(f)  1 0 1 0 1 0 1 0 1\¨⊂[1]W        ?

(g)  1 0 1 0 1\¨⊂[2]W                ?

(h)  ⊃[2]1 0 1 0 1 0 1 0 1\¨⊂[1]W    ?

(i)  ⊃[1]1 0 1 0 1\¨⊂[2]W            ?
```

Technically "backslash" is an operator — a decision made to remove the ambiguity of this symbol in APL1. Operators may have either arrays or functions as *operands* (n.b. not arguments — these relate only to functions), and so the expand function of APL1 is now thought of as "backslash deriving expand". However this distinction creates no problems in exercising enclose, disclose and each, since in all cases we are dealing with a derived function "1 0 . . . 1-expand".

And now the answers with comments?

(a) No APL2 involved — inserts blanks between existing columns:

```
A N D                                  DR = 1 5 5
B O Y
C A N
D A D
E A T
```

(b)

```
┌───┐ ┌───┐ ┌───┐ ┌───┐ ┌───┐          DR = 2 5
│AND│ │   │ │AND│ │   │ │AND│
│BOY│ │   │ │BOY│ │   │ │BOY│
│CAN│ │   │ │CAN│ │   │ │CAN│
│DAD│ │   │ │DAD│ │   │ │DAD│
│EAT│ │   │ │EAT│ │   │ │EAT│
└───┘ └───┘ └───┘ └───┘ └───┘
```

Alternate elements are 5x3 blocks of space.

(c)

```
┌─────┐ ┌─────┐ ┌─────┐          DR = 2 5 3
│A A A│ │N N N│ │D D D│
└─────┘ └─────┘ └─────┘
┌─────┐ ┌─────┐ ┌─────┐
│B B B│ │O O O│ │Y Y Y│
└─────┘ └─────┘ └─────┘
┌─────┐ ┌─────┐ ┌─────┐
│C C C│ │A A A│ │N N N│
└─────┘ └─────┘ └─────┘
┌─────┐ ┌─────┐ ┌─────┐
│D D D│ │A A A│ │D D D│
└─────┘ └─────┘ └─────┘
┌─────┐ ┌─────┐ ┌─────┐
│E E E│ │A A A│ │T T T│
└─────┘ └─────┘ └─────┘
```

"Each" causes the derived function to be applied to all 15 elements of W, and then sews the resulting 15 strings back into a 5x3 structure.

(d)

```
┌─────┐ ┌─────┐ ┌─────┐ ┌─────┐ ┌─────┐
│ABCDE│ │     │ │NOAAA│ │     │ │DYNDT│ DR = 2 5
└─────┘ └─────┘ └─────┘ └─────┘ └─────┘
```

First, rows lose their identify, i.e. spacing is applied to the vector of 3 columns.

(e)

```
┌───┐ ┌─┐ ┌───┐ ┌─┐ ┌───┐ ┌─┐ ┌───┐ ┌─┐ ┌───┐
│AND│ │ │ │BOY│ │ │ │CAN│ │ │ │DAD│ │ │ │EAT│  DR = 2 9
└───┘ └─┘ └───┘ └─┘ └───┘ └─┘ └───┘ └─┘ └───┘
```

This time columns lose their identity and spacing is applied to 5 rows of W.

(f)

```
┌─────────┐ ┌─────────┐ ┌─────────┐
│A B C D E│ │N O A A A│ │D Y N D T│     DR = 2 3
└─────────┘ └─────────┘ └─────────┘
```

As for (d) but now the spacing is applied internally to the columns.

(g)

```
┌─────┐ ┌─────┐ ┌─────┐ ┌─────┐ ┌─────┐
│A N D│ │B O Y│ │C A N│ │D A D│ │E A T│ DR = 2 5
└─────┘ └─────┘ └─────┘ └─────┘ └─────┘
```

As for (e) but spacing is applied *internally* to the columns.

```
A B C D E                          DR = 1 3 9
N O A A A
D Y N D T
```

Internal structure of (f) appears as last element of shape vector.

```
ABCDE                              DR = 1 5 5

NOAAA

DYNDT
```

Internal structure of (g) appears as first element of shape vector.

SUMMARY

The secret of mastering APL2 is to take things in easy stages and to enhance the formal vocabulary of concepts with one's own informal vocabulary. In the preceding pages you will have seen examples of this applied to the basics of enclose, disclose and each. Given the indulgence of the editors of 'VECTOR', a later issue may provide a similar path through the (not so) arcane mysteries of operator extension!

OPERATORS AND NESTED ARRAYS IN DYALOG APL

by John Scholes

Abstract

Second generation APLs such as Dyalog and IBM's APL2 allow two conceptually simple extensions to the APL language. Arrays may contain further arrays as elements; an operator may combine with any function to produce a derived function. This paper discusses the power which these extensions add to the language.

Nested or enclosed arrays, and the extension of the domain of operators 'invent' themselves quite naturally in APL. People — particularly those learning APL, will often type expressions which seem reasonable but which give errors. If, for instance

```
        A←3 3 3
```

works, why doesn't

```
        B←3
        A←B B B
```

Similarly, if

```
        A,B,C
```

works, why doesn't

```
        ,/A B C
```

We can draw an analogy between nested arrays in APL and 'imaginary numbers' in mathematics. If the polynominal $x^2 - 1 = 0$ has solutions then why doesn't $x^2 + 1 = 0$? The introduction of 'i' the imaginary square root of ¯1 brought a completeness to this area of maths and found immense application in a wide range of very practical fields. So nested arrays bring a completeness and power to APL.

APL has always had recursive functions — now we have recursive data structures to complement them. Many problems have a natural expression in nested structures, for instance, representation of hierarchy in organisations or modelling of products made from sub-assemblies. These problems have in the past been modelled with function structures which match those of the problem, but with at best, clumsy data representation.

Enclosed arrays may 'invent' themselves in many ways in APL. As one example, if we type ,/1 2 3 what does the result look like? Firstly, it must be a scalar because the reduction of a vector reduces the rank by one, and secondly, the value of this scalar must be the concatenation of the numbers 1, 2 and 3. We have a scalar whose single value is a vector — an enclosed vector. Similarly, the catenate reduction of a matrix would produce a vector of enclosed vectors.

From the introduction of this single new data type many of the properties follow. If we 'comma reduce' a vector of vectors, we produce a single scalar, whose element is a vector, each of whose elements is a vector — arrays may be nested to arbitrary depths. Our vector of vectors may be: reshaped; catenated with other conforming arrays; rotated; and in fact, subjected to all of the structure-primitive functions. Using partial specification of the form

```
A[...]←B
```

we can replace any element of an array with any other array. Nested arrays may therefore have arbitrary shape, rank and depth. Of course, individual implementations enforce limits on depth, rank and shape of arrays, but these limits are usually no more of a hindrance to working than their equivalents in first generation APLs.

ANNOUNCING

dyalog APL

VERSION 3·0

The HIGH PERFORMANCE APL for UNIX* systems

Dyalog APL is FAST

The new version of Dyalog APL is at least twice as fast as before. The component file system and the performance of many primitive functions has been substantially enhanced, so that for some applications even greater improvement will be experienced. Our own benchmarks based on real applications indicate that there is now no faster APL available for supermicros.

Dyalog APL is FUNCTIONAL

Features include nested arrays, full-screen editor, full-screen data manager, component file system, error trapping, external variables, large dynamic workspace.

Dyalog APL is POWERFUL

Dyalog APL's interface to the UNIX* Operating System is unrivalled. Not only can you run any UNIX*program from within the workspace, but you can capture its results in an APL variable for subsequent processing. Furthermore, you can write functions in a compiled language such as C or Fortran.

Dyalog APL is RELIABLE and WELL ESTABLISHED

Dyalog APL is now implemented on over 20 different UNIX*systems and has already been installed in more than 50 companies in eight countries.

For further information contact

Sales Department
Dyadic Systems Limited
30 Camp Road Farnborough Hampshire GU14 6EW
Telephone Farnborough (0252) 547222 Telex 858811

*UNIX is a trademark of A T & T Bell Laboratories

In particular, a special case in an array all of whose elements happen to be scalars — these are the simple arrays of first generation APLs, with one extension — we no longer have the constraint that all of the elements have to be of the same type — all numbers or all characters — it is now perfectly valid to type

```
        A←1 2 3,'ABC'
```

to form a simple six element vector — we can do arithmetic with the first 3 elements

```
        +/3↑A
```

but get a DOMAIN error if we include character elements

```
        +/4↑A
```

— each element of an array has its own property of type (numeric or character) rank and shape.

How should the scalar functions act on our now nore general concept of an array? If A is ,/1 2 3. what is 10 + A. The result of adding scalar 10 to scalar A must yield a scalar, and the most reasonable guess (which is in fact the case) for its value, is the scalar whose element is vector 11 12 13.

What about (1 2 3)4 + 10(20 30 40)? This must mean the addition of two two-element vectors — the juxtaposition of two arrays without an intervening function has always signified a vector although in the past this was only meaningful for scalar literals of the same type. From the rules of scalar conformability, the result must be a two element vector whose first element is (1 2 3) + 10 and whose second is 4 + (20 30 40) so the result must be 11 12 13 24 34 44.

So the scalar functions 'pervade' to any depth of array, providing that at each level, the arrays are scalar conformable.

Operators

In order to give examples, first let's see a new primitive operator called "each". "Each" takes its functions and applies it to (or between) each element of its argument(s) following the rules of scalar conformability.

```
              ⍴¨(1 2 3)(4 5)
        3  2
                   2 3⍴¨4 5
               4 4  5 5 5
```

The "each" operator appears so powerful and natural that ALL of the second generation APLs have included it as a primitive.

Now let's look at the other implications of being able to do,/1 2 3. The advent of nested arrays enables us to extend the domain of operators. In first generation APLs, operators could represent results only for functions which produced scalar results from scalar arguments. Now that this restriction is lifted operators may apply ANY functions.

Defined Functions

```
      ∇ Z←A ADD B
[1]     Z←A+B
[2]     A,'+',B,'→',Z
      ∇
```

```
          3 ADD 4
3 + 4 → 7
7
```

```
      ADD/1 2 3 4
3 + 4 → 7
2 + 7 → 9
1 + 9 → 10
10
```

If, for instance, PERSONNEL were a vector of information — one element per person — and PRINT, a defined function to print information for one person:

```
      ∇ PRINT INFO
[1]     'NAME:        ',1 ⊃ INFO
[2]     'AGE:         ',2 ⊃ INFO
      ∇
```

Then `PRINT¨PERSONNEL`

would print the information for each person.

Derived Functions

Operators combine with functions to form derived functions, these in turn may be applied by operators.

```
      A←((1 1)(2 2)(3 3)) ((4 4 4)(5 5 5))
      ρA
2
      ρ¨A
 3  2
      ρ¨¨A
  2  2  2   3  3
      +/¨¨A
 2 4 6  12 15
      ADD/¨A
2 2 + 3 3 → 5 5
1 1 + 5 5 → 6 6
4 4 4 + 5 5 5 → 9 9 9
  6 6     9 9 9
```

Conclusion

Already we see examples of pieces of code contrasting first and second generation APL, in much the same way as we used to see examples contrasting APL and BASIC or PASCAL, for example:

```
      ∇ LIST1;NL
[1]    ⍝ VECTOR REPRESENTATION OF ALL FUNCTIONS
[2]     NL←⎕NL 3
[3]    L:→(0∊1↑⍴NL)/0
[4]     ⎕VR NL[⎕IO;]
[5]     NL← 1 0 ↓NL
[6]     →L
      ∇

      ∇ LIST2
[1]    ⍝ VECTOR REPRESENTATION OF ALL FUNCTIONS
[2]     ⎕∘←∘⎕VR¨↓⎕NL 3
      ∇
```

Nested arrays and extended operators have been available in APL for only a short time, but the increased power that they bring to the language means they are here to stay.

APL SOFTWARE TECHNOLOGY (UK) LIMITED

SPECIAL PRICE OF 15K

Yes, 15K! That is the small price for **Powertools/PC**, *the software product which will transform your APL*PLUS/PC workspaces, with:*

- *A wealth of useful function*
- *Matrix catenation, indexing and membership*
- *String handling, super boxing*
- *Sub-string searching and indexing*
- *Booleans implemented as bits rather than words*

PLUS

The superb **Forms Processor**:

- *Multiple forms that may be larger than the screen*
- *Inbuilt data validation*
- *APL function execution at the field level*
- *For the developer Form definition and editing with semi-graphic drawing facilities and interactive Forms testing*

Powertools/PC *is written in Assembler, is really fast and CPU efficient.*

Powertools/PC *has been specifically designed to act as a base around which the application system can be built. It is NOT the typical AP124-type screen manager.*

All this for just 15K (plus or minus the odd bit here and there) and that will soon be repaid by way of freed workspace.

Powertools/PC, *the workspace and CPU saver, is just £295 per copy (excluding VAT).*

CONSULTANCY

Our APL consultancy and support services are available on mainframes, minis and micros. Specific expertise in the financial and insurance industries.

We are also very interested in talking to any APL consultants who are looking to further their careers in new technology.

ERICSSON

We can supply **Ericsson's** *superb fully IBM/PC compatible micro*

PC or XT versions *128 to 640KB RAM*
mono or colour monitor

Ericsson's *newly announced* **Portable PC** *micro is now available*

You don't have to lug this one
Full size Plasma screen *Up to 512KB RAM*
Built in printer (optional)

*We can also offer our own APL machine, an Ericsson XT complete with 512 or 640KB RAM and the 10mhz 8086, plus a copy of APL*PLUS/PC release 4.*

14 Rosewood Avenue, Alveston, Bristol BS12 2PP. Tel: 0454 415737.

POWER TO APL

by Per Hultin and John Hagger

APL has come a long way since its inception many years ago, more years ago than we care to remember, not only in the enhancements and language extensions, and number of product offerings, but in the size of machines that can now support APL. It was not that long ago that DP managers for large mainframes were known to cringe whenever an APL user threatened to log on, but now implementations are available on micros the size of the QL. No longer is the number crunching mainframe required to support the number one programming language, but small relatively inexpensive micros such as the IBM and compatible PCs can be used with powerful efficient APL offerings.

However, anyone that uses these smaller machines knows that large amounts of computer capacity can be tied up performing relatively mundane activities such as formatting and neat presentation of data to the user. If data structures are large, bottlenecks are created in working with that data.

APL certainly is great for getting to the heart of the problem solving activities, but the interface to the user by way of screen and report presentation usually accounts for a large percentage of the program code, and normally for the CPU comsumption too.

We saw that there was a requirement to tackle these areas of user to APL interface, and excessive CPU consumption, to provide a powerful tool-box of useful but efficient functions and facilities, together with a simple but unique approach to the problem of program to user interface. We also saw APL as not being the true answer to this problem. We did not think that the efficiency problem could be solved by the usual 'box of tricks' solution, or by providing the typical fullscreen management facility (like VS APL's AP124) to provide for the program to user interface, this just replacing one nightmare by another.

The only really efficient coding technique that could solve these problems seemed to be Assembler, but to use this problem of interfacing APL to Assembler needed to be solved. This allows the Assembler code to be external to the APL interpreter, but utilises simple call structures and allows access to the symbol table to provide for variable reading and creation. Only by having the code external could the product be made really compact, and enable easy upgrading, for instance in forthcoming network environments.

The first step in the development was to generate the useful functions and evaulate the effectiveness of such a solution. We decided on the types of facilities required and broadly grouped these into:

— matrix catenation
— matrix index / membership
— string handling
— boolean handling

The boolean handling functions were a special case to us. We had decided to base our solution upon STSC's APL*PLUS/PC offering, and in the current implementation boolean bits are actually stored internally as integer word zero and ones, consuming vast amounts of storage when large data structures are being handled. A complete 'compressed boolean' operation set would dramatically affect any workspace utilising it.

Early developments showed that for many of the facilities implemented CPU consumption was something in the order of 10% of the APL equivalent solution. Preparation of data for presentation in report form to the user was now fast and effective, allowing the response time to be the domain of the problem solving code rather than to be extended by the user interface coding.

The second development stage was that of the Forms Processor. Here we were not looking for the standard AP124 Full Screen Manager solution, but a logical extension of the problem program. The program needs to create data that is presented to the user for his attention and possible modification or entry of new data. Restrictions such as screen size, data validation, incorrect keying should not be the constant worry of the developer, but of the user/program interface package, allowing him to concentrate on the real job in hand, that of problem solving.

We decided that the salient features of such a facility should be:

- Provision of up to 64K forms buffer into which a maximum of 31 forms could be stored, the number being dictated by the size of the forms in use. This buffer could be either the video memory (depending on size) or an APL variable.
- Display of any of the forms stored onto variable screen windows, allowing the user to 'drag' the form under the current window.
- The forms should have fields, just as in AP124, with associated attributes, and no size restrictions.
- Support of various data formats, character matrix, fixed length fields or vector with field delimiters. Fields to be declared as character, numerics, with or without decimals, dates, etc., as required entry, fixed length, or multiple entry, etc., with the Forms Processor performing checks accordingly.
- APL functions to perform error checking or deriving data at field level, or any of the functions to be executed triggered by the change of a particular field.
- Overlapping fields, allowing the user to consider the fields as separate, but the APL function to treat them as an entity.
- Support for self sustained form overlay structures that could, if required, be held as components on a file.

These should combine to firm an effective solution to the problem of interfacing data to the user.

Evaluations showed that the screen handling was very fast, with movement, error checking, data validation, etc, being, to the user, instantaneous. Any time delay could be directly associated with the problem solving, and not the data display/user interface code.

We are of no doubt that APL has a tremendous and growing future, and that by working with it, and not against it, one can produce dynamic user-acceptable systems. To do this APL needs that added power that can be provided by tackling of specific areas of concern.

The solution lies in assisting APL in the areas it does not best suit, and combining the developments into an enhanced APL offering. In this way we shall see more and more APL systems on small desk top systems with users aware that APL provides a fast and effective solution to their problem solving requirements.

INDEX TO ADVERTISERS

	Page
APL People Ltd	57
APL Software Technology	122
Cocking & Drury Ltd	20
Dyadic Systems Ltd	118
HMW Programming Consultants Ltd	58
IBM Sweden	2
Mercia Software	16
Mercia Software	80
MetaTechnics Systems Ltd	106
MicroAPL Ltd	23
MicroAPL Ltd	90
A Smith	89

Contents of Volume 1

Title	*Author*	*Issue*	*Page*
1984 IPSA Users Meeting	Adrian Smith	3	65
4GLs Killing APL?	Desmond Booth	2	49
A DIF file interface	Romilly Cocking	4	149
A Graphic Vision of APL	Graeme Robertson	1	36
An APL Compiler	Roy Sykes	2	54
An APL Dialogue	Graeme Robertson	3	135
APL2	Jim Brown	2	56
APL84 in Focus	Smith, Bowman & Lyus	2	52
APL and GKS	Richard Nabavi	4	63
APL and Graphics	Adrian Smith	1	35
APL and Graphics Standards	Richard Bowman	2	75
APL and Partitioned Data	Johnathan Barman	1	129
APL in the Information Centre	Adrian Smith	1	41
APLpip	Llewellyn Jones	2	109
APL vs packages and 4GLs	Adrian Smith	2	48
BAA 1983 Report and Accounts	Johnathan Barman	1	5
Case Study: Imperial Group plc	Adrian Smith	2	89
Communicating with APL	Adrian Smith	1	48
Complex numbers in APL	Alan Hawkes	1	107
Duck a la carte	Peter Donnelly	4	65
Dyalog APL	John Scholes	2	54
Expert Systems	M.J. Winfield	3	49
FGL: Fifth Generation Language	Robert Bittlestone	1	89
Generic Read and Replace in IPSA	Robert Pullman	3	128
Getting the best out of GDDM	Adrian Smith	4	65
GKS: an Opportunity for APL	Richard Nabavi	2	78
Graphics for Decision Makers	David Preedy	2	83
IBM-based APL Communications	Tim Perry	1	62
Inside the international APL standard	David Ziemann	1	143
Interfacing Viewdata and APL	John Pym	1	59
ISO Standards update	David Ziemann	2	125
Keyword QL/APL review	David Ziemann	4	121
Matchmakers: a simulation case study	Adrian Smith	1	80
Mr Babbage's Secret: A presentation	Ole Franksen	2	61
Moving Data from 1-2-3 to APL	Michael Carmichael	4	130
Networking APL Microcomputers	Richard Nabavi	1	49
Numeric Integration and Prob. Dist.	Alan Hawkes	2	117
PATTIE: a practical Expert System	Christopher Harvey	3&4	101&98
Preferring a Menu-oriented Dialogue	Gosta Olavi	3	83
QL/APL - A new popularisation of APL	David Eastwood	4	81
Quotation Jottings on Matters Temporal	Michael Carmichael	3	156
Random Contingency Tables	Dr. A.G. Prys Williams	3	160
Reinventing Man	Robert Bittlestone	4	75
SCREENIO:An IBM Full Screen Manager	David Doherty	1	121
Setting up a Company Info. Centre	Romilly Cocking	1	42
Statistical computing with APL	Alan Hawkes	2	113
Steps towards a better BASIC	Anthony Camacho	1,2&4	77,73&95
The Ampere: a Portable APL Machine	Philip Van Cleave	2	55
The APL Machine	Jim Ryan	2	57
The Global Information Centre	Fred Perkins	1	45
The Information Centre as a strategy	Fred Perkins	4	90
The paper they dared not print	Anonymous	1	87
Towards a Teachable Interface	Adrian Smith	3	91
What's New for 85	Dick Bowman	3	63
When domino is not sufficient	Sykes and Ansell	4	155
Why APL? A non-technical introduction	Robert Bittlestone	1	65
XPL: Expert Systems Framework in APL	Robert Bittlestone	2	65

BACK NUMBERS OF VECTOR

Back numbers of VECTOR are available from the BCS. If you don't have them all, now is the time to complete your collection. There is an index to volume one in this issue of VECTOR to tempt you. Apart from the technical contents every issue includes book and product reviews, letters, news and a competition. Send in your order before they run out. These will one day be unobtainable collectors' items like the early issues of Quote Quad.

The prices inclusive of postage and packing are as follows:

	Prices in Pounds Sterling		
	UK	*Surface (inc. Europe)*	*Airmail (outside Europe)*
Single issues	3	3.75	5.75
Volume 1	10	14.00	22.00

Please send sterling cheques or money orders payable to The British APL Association to:

The British APL Association, 13 Mansfield Street, London W1M 0BP

Don't forget to include your name and address and to be clear which VECTORs you want.

BRITISH APL ASSOCIATION

Membership Application Form

Please read the membership information in the inside front cover of VECTOR before completing this form. Use photocopies of this form for multiple applications. The membership year runs from 1st May 1985 — 30th April 1986.

Name:	
Department:	
Organisation:	
Address line 1:	
Address line 2:	
Address line 3:	
Address line 4:	
Post or zip code:	
Country:	
Telephone number:	

Membership category applied for (tick one):

Non-voting student membership	£ 5
UK private membership	£ 9
Overseas private membership......	£ 16
Airmail supplement (not needed for Europe)	£ 8
Corporate membership	£ 75
Corporate membership Overseas ...	£120
Sustaining membership	£325

For student applicants:

Name of course:	
Name and title of supervisor:	
Signature of supervisor:	

PAYMENT

Payment should be enclosed with membership applications in the form of a UK sterling cheque or postal order made payable to "The British APL Association". Corporate or sustaining member applicants should contact the Treasurer in advance if an invoice is required. Please enclose a stamped addressed envelope if you require a receipt.

Send the completed form to the Treasurer at this address:

Mel Chapman, 12 Garden Street, Stafford, ST17 4BT, UK.

THE BRITISH APL ASSOCIATION

The British APL Association is a Specialist Group of the British Computer Society and a member of EuroAPL, an organisation supported by the Commission of the European Communities. It is administered by a Committee of eight officers who are elected by the vote of Association members at 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.

Photoset by Wizard Printing Services, Brookwood, Surrey.
Printed in England by Copyprint Ltd., London SW1.

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 and a member of EuroAPL, an organisation supported by the Commission of the European Communities. APL stands for "A Programming Language" — an interactive computer programming language noted for its elegance, conciseness and fast development speed. It is supported on many timesharing bureaux and on most mainframe, mini and micro 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.

APL People	17 Barton Street Bath, Avon Tel. 0225-62601
APL Software Technology	14 Rosewood Avenue, Alveston, Bristol, BS12 2PP Tel. 0454-415737
Cocking & Drury Ltd.	16 Berkeley Street, London W1X 5AE Tel: 01-493 6172
Dyadic Systems Ltd.	30 Camp Road, Farnborough, Hants Tel: 0252 547222
Inner Product Ltd.	Eagle House, 73 Clapham Common Southside, London SW4 9DG. Tel: 01-673 3354
MetaTechnics Systems Ltd.	Unit 216, 62 Tritton Road, London SE21 8DE Tel: 01-670 7959
MicroAPL Limited	Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London SW8 5BP Tel. 01-622 0395
I.P. Sharp Associates	10 Dean Farrar Street, London SW1H 0DX. Tel. 01-222 7033

The British Computer Society, 13 Mansfield Street, London W1M 0BD.