VECTOR

100+ PAGES OF THE BEST IN APL

- Improve your fibbing.
- APL System design – rules of thumb.
- The APL debates.
- New awards announced.
- The latest UK and international APL news

The Journal of the British APL Association

A Specialist Group of the British Computer Society

Vol.3 No.4 April 1987

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. Authors are requested wherever possible to supply copy in machine-readable form ideally text files on a 5¼" IBM-PC compatible diskette. For other standards, 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 Rates 1986-87

Category	Fee p.a.		VECTOR copies	Passes
	£	$		
Nonvoting student membership	5		1	1
UK Private membership	10		1	1
Overseas private membership	18	27	1	1
Supplement for airmail (not needed for Europe)	8	12		
Corporate membership (UK)	85		10	5
Corporate membership (Overseas)	140	210		
Sustaining membership	360		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 may be paid in £UK or $US.

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

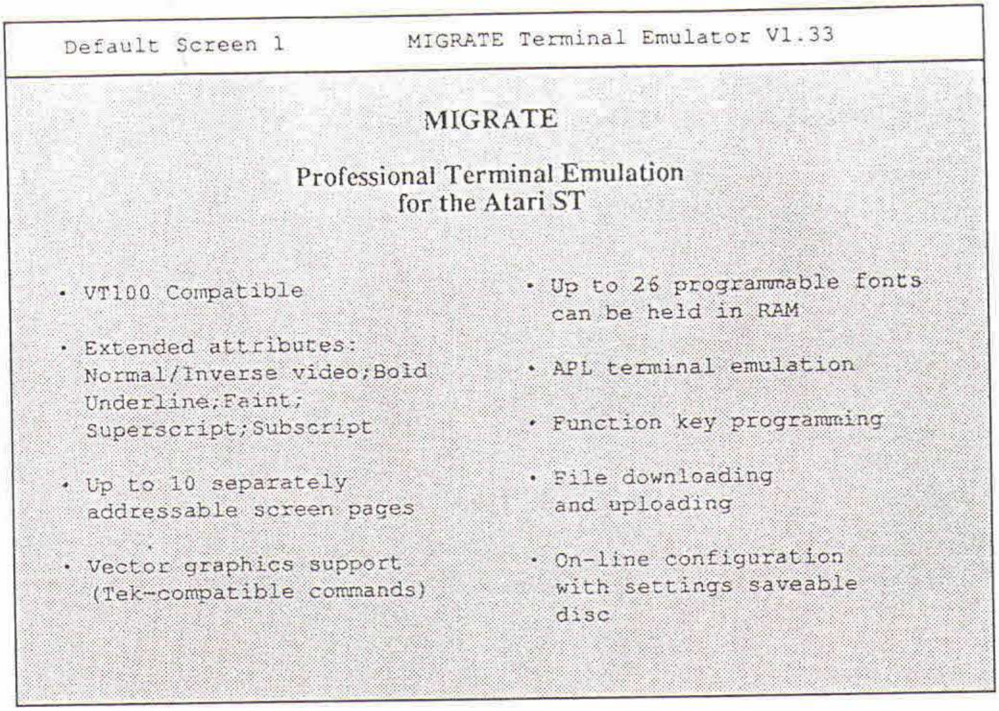
Advertisements in VECTOR should be submitted in typeset camera-ready A5 portrait format with a 20 mm blank border. Illustrations should be black-and-white photographs or line drawings. Rates are £250 per page. A6 and A7 sizes are offered subject to layout constraints.

Deadlines for advertisement bookings and receipt of camera-ready copy are given beneath the Quick-Reference Diary.

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

Vol.3 No.4

Contents



```
Default Screen 1          MIGRATE Terminal Emulator V1.33
```

MIGRATE

Professional Terminal Emulation
for the Atari ST

```
• VT100 Compatible                • Up to 26 programmable fonts
                                    can be held in RAM
• Extended attributes:
  Normal/Inverse video;Bold       • APL terminal emulation
  Underline;Faint;
  Superscript;Subscript           • Function key programming

• Up to 10 separately             • File downloading
  addressable screen pages          and uploading

• Vector graphics support         • On-line configuration
  (Tek-compatible commands)         with settings saveable
                                    disc
```

MIGRATE is a powerful terminal emulator program for the Atari ST computer. MIGRATE will operate on any model in the ST range, provided it is fitted with a high resolution monochrome monitor. MIGRATE offers a wide range of features that allow the Atari ST to be used as a low-cost, high performance terminal emulator for many different host computer systems. MIGRATE is available on a single sided disc together with a comprehensive reference guide and sample fonts for £99 plus VAT from:

MicroAPL Limited

Unit 1F Tideway Industrial Estate
87 Kirtling Street
London SW8 5BP
Telephone: 01-622 0395

Telex: 896885 IOTA

Editorial: 3 years on

by David Preedy

As this will be my valedictory contribution as editor of VECTOR it seems appropriate to take stock of APL's current position, assess what the APL community has achieved over the recent past and anticipate the major challenges over the coming years.

The past few years have seen several landmarks – the conference at Loughborough, the launch of VECTOR, the "Best of APL 84" seminar – culminating in last year's APL 86 Conference at Manchester. The success of that conference can be gauged by the supporting material produced, with the normal Proceedings supplemented by a Tutorials volume and by the various reports on debates published in recent issues of VECTOR.

These events have been highly visible, but should not detract from the wealth of background work to which many in the APL community have contributed. There have been major technical developments bringing APL interpreters onto a widening range of machines, producing an APL compiler and extending the language itself into such realms as nested arrays, user-defined operators, and the like. There has been a wealth of innovative applications (far too many of which remain under wraps), and a growing extension of the use of APL in the bastions of higher education.

However, despite the B.A.A. being one of the thriving specialist groups of the British Computer Society, APL is still an elitist language, largely unknown in the computing community and widely misunderstood in those areas where it is known. The membership of the B.A.A. may have grown over the past few years, but most of us believe that it remains a small fraction even of those who use APL – at least as measured by sales of APL interpreters – and it must be a minuscule percentage of those who could derive a significant benefit from using APL.

On the education front, the majority of students on computer-related courses at Universities and Polytechnics graduate without having heard of APL, let alone used it in anger, and the position in the schools is even worse despite the widespread availability of microcomputers and the evident suitability of APL, together with other languages such as LOGO, for educational work.

The public image of APL has barely changed over the past three years. APL is largely ignored by most of the British computing press. It is symptomatic that only two publications (VECTOR excluded of course!) mentioned the APL86 conference, even though it was the first time that APL's international event had been held in the U.K. The articles that do appear show little sign of a growing understanding of APL – they still tend to comprise the standard contents of an introduction to APL explaining how miraculous it is that 1 + 1 can still produce the answer 2 and then introducing a hand-full of the amazing APL hieroglyphs; my own favourite is the x symbol that I was taught when I was eight or nine but had to un-learn in order to tell computers to multiply!

However, times have changed! We are now standing on a threshold of opportunity created by the achievements of the past years. The B.A.A. accounts for 1986/87 will show the scale of the success generated by the hard work of all who helped organize APL86, but it can be no secret that part of the reward has been a substantial profit for the B.A.A. Since then the B.A.A. committee has been actively drawing up plans to ensure that the proceeds of the conference are directed towards the long-term interests of the APL community.

Various schemes have been proposed and elsewhere in this issue of VECTOR we announce two which have been approved – an Outstanding Achievement award to recognise those who have made a significant contribution to the advancement of APL; and a plan to provide financial backing to proposed projects.

However the most significant challenge must surely be how to extend an awareness of APL into the schools. Computing within education is moving out of its phase of development. Today's methods, based around the dominant position of BASIC, are becoming entrenched as best practice. It will soon be too late to establish a new approach incorporating widespread use, not only of APL, but also of other languages such as LOGO, which place the interests of the user, in this case the schoolchildren, above those of the computer.

It is for these reasons that the B.A.A. has become one of the major backers of the separately established I-APL project to develop a free interpreter targetted towards schools. The B.A.A.'s contribution to this venture has been crucial and I am sure that the committee will have the foresight to continue supporting the project through to its fruition.

Of course these plans require more than just money; to be fully successful they need the whole-hearted and active support of those of us already lucky enough to appreciate the advantages of APL. The I-APL team will carry out the technical development of the interpreter; they are already fully immersed in the preparation of the supporting material, particularly that aimed at showing teachers how APL can be used most effectively in the classroom; they can also develop a marketing plan to spread the word among the teaching profession.

However they are too small a group to be able to spend a couple of hours in front of a P.C. with each teacher interested in finding out about APL. This is precisely the sort of role where every one of us can help. If we can rapidly build up a nucleus of schools where APL is in use, the word will go around the teachers' grapevine, and a chain-reaction will develop. There is then a realistic chance that our own children's first experiences of computers will encompass more than the constraints of BASIC.

Quick-reference diary

compiled by David Preedy

Date	Venue	Event
1987		
10-14 May	Dallas	APL 87 – APL in transition
5 June	London	British APL Association AGM & meeting
8-11 September	Strasbourg	European Software Engineering Conference organised by AFCET, Paris
18 September	London	British APL Association meeting
16 October	London	British APL Association meeting
20 November	London	British APL Association meeting
1988		
15 January	London	British APL Association meeting
1-5 February	Sydney	APL88 – "APL – Past, Present, Future"
18 March	London	British APL Association meeting
20 May	London	British APL Association AGM & meeting
16 September	London	British APL Association meeting
21 October	London	British APL Association meeting
18 November	London	British APL Association meeting

All British APL Association meetings are to be held at the Royal Over-Seas League, Park Place, near Green Park tube station and start at 2pm.

Please note the changed date of the A.G.M., which has been re-scheduled to avoid a clash of dates with APL87 in Dallas.

Dates for future issues of VECTOR

	Vol 4 No 1	Vol 4 No 2	Vol 4 No 3	Vol 4 No 4
Copy date	24 Apr 87	24 Jul 87	16 Oct 87	29 Jan 88
Ad. booking	22 May 87	21 Aug 87	13 Nov 87	19 Feb 88
Ad. copy	29 May 87	28 Aug 87	20 Nov 87	26 Feb 88
Distribution	July 87	October 87	January 88	April 88

Enhancing APL.68000

An intriguing computer for an exciting language—the WS-1 and APL.68000. At last the APL programmer can have portability without sacrificing power or capability. Dodge the queue waiting for time on the mainframe and discover the sudden freedom of being able to try out programs anytime, anywhere.

The APL.68000 interpreter is implemented in 86KB of ROM, running under a multi-user, multi-tasking operating system called BIG. DOS. Speed is the essence of APL programming, and now the WS-1 makes development even faster.

Come and see us at APL '86.

APL.68000 on the WS-1 has attractive enhancements such as a powerful component file system, QUAD. FMT function for alpha report formatting, QUAD. CC function for full-screen control, and extended error trapping facilities.

Bundled with the WS-1 are four workspaces: SYSFNS, APLUTIL, FILEUTIL, and SYSCOM. Each gives access to the WS-1's unique capabilities such as control of the built-in speaker phone, microcassette unit, RTC (real time clock), bit-mapped graphics LCD screen, and optional 3.5-inch floppy disk drives.

Compress these capabilities into a sleek footprint measuring less than 13 inches by 11 inches, and you have the ultimate definition of power.

ampère

FOR DISTRIBUTORSHIP INFORMATION AND PRODUCT DETAILS PLEASE CONTACT:

ampère INCORPORATED Asahi Bldg., 7-5-20 Nishi-Shinjuku, Shinjuku-ku, Tokyo, Japan. *Phone:* 03-365-0825. *Telefax:* 03-365-0999. *Telex:* J33101 AMPERE. IP Sharp Mail Box Code AMP (Group Code APLWS).

APL course diary

Many of the APL vendors included in the VECTOR APL Product Guide offer courses in APL and related topics. For a full list readers are recommended to look under the relevant section of the product guide. This section gives course dates for those suppliers who have prepared their course schedule at the time of going to print.

April 1986		
7-9	APL Fundamentals	Cocking & Drury
21-22	APL*PLUS/PC Enhancements	Mercia Software
28-30	APL Fundamentals	Cocking & Drury
May 1986		
5-6	APL*PLUS/PC Enhancements	Mercia Software
11-14	APL System Design	Cocking & Drury
12-14	System design with APL*PLUS	Mercia Software
19-21	APL Fundamentals	Cocking & Drury
19-21	APL*PLUS/PC Introduction	Mercia Software
27-28	Statgraphics	Cocking & Drury
June 1987		
9-11	APL Fundamentals	Cocking & Drury
July 1987		
7-9	APL Fundamentals	Cocking & Drury
14-15	Statgraphics	Cocking & Drury
28-30	APL Fundamentals	Cocking & Drury

APL Courses

APL Fundamentals	April 7 – 9	£375
APL Fundamentals	April 28 – 30	£375
System Design	May 11 – 14	£595
APL Fundamentals	May 19 – 21	£375
Statgraphics	May 27 – 28	£240
APL Fundamentals	June 9 – 11	£375
APL Fundamentals	July 7 – 9	£375
Statgraphics	July 14 – 15	£240
APL Fundamentals	July 28 – 30	£375

Discounts are availale to companies making more than one booking at a time. All prices exclude VAT. To book, or for further details, contact Beverley Satterley at the address below.

COCKING & DRURY LTD.
THE APL PROFESSIONALS

16 Berkeley Street, London W1X 5AE

Tel: 01 – 493 6172

General Correspondence

The VECTOR working group welcomes correspondence on any topic affecting the APL community. All such letters should be addressed to the Editor and should indicate whether they are intended for the general or the technical section. Letters containing APL code will normally appear in the Technical Section of VECTOR, and authors are asked to observe the requirements on the inclusion of APL code stated on the inside cover. The Editor reserves the right to edit any letter unless the writer states that it is to be published in full or not at all.

APL takes wing

From Mr Norman Thomson 24th December 1986

Sir,

I found the following as part of the results of a recent bibliographic search on APL.

> "Bird-borne satellite transmitter and location program" Strikwerda, Fuller, Seegar, Howey
>
> Several birds carrying APL-developed transmitters have been tracked by satellite with good results, but more work should have been done to develop lighter, more reliable hardware.

Yours,

Norman Thomson,
17 St James Terrace,
Winchester, Hants., SO22 4PP.

P.S. I understand that APL is the Applied Physics Lab at John Hopkins University.

(Editor: It sounds as though this development puts our laptop systems in the shade!)

APL 87 in Dallas

From Mr Jim Fiegenschue 15 December 1986

Sir,

I am writing to tell you about the upcoming international APL conference, APL87, which I think will be of interest to your readers.

I thoroughly enjoyed APL86 in Manchester; the B.A.A. has set a very high standard of excellence for us to try to match!

Yours truly,

Jim Fiegenschue,
Dallas

(Editor: The details Jim sent are included in the International APL News section of this issue of VECTOR.)

APL Publicity

From Dr Peter Branson 20th January 1987

Sir,

I enclose a copy of an article as it appeared in PC Week on 26th November 1986. There were several typos in the code. (I sent camera-ready copy, but, of course, they re-set it!) The text however is pretty faithful to the original except that "insufficiently known" for APL becomes "sufficiently known".

The APL technical content is deliberately low, but at least it got published (with alacrity – somewhat to my surprise). There were perhaps a couple of reasons for this:

(1) a careful study of previous issues of PC Week gave me several hooks to latch on to and attract the editor's interest;

(2) I photocopied bits of everything I referred to, so that he would have no trouble checking accuracy.

It won't always work, but perhaps the above points could be commended to other VECTOR readers struggling with the hard job of publicising APL.

Yours faithfully,

Peter Branson,

Oaklands Cottage,
Wray Common,
Reigate, Surrey.

(Editor: I hope that other budding authors will bear your points in mind – and perhaps let us publicise other tricks of the trade. This may be especially relevant in the light of the proposed B.A.A. prizes, announced in the B.A.A. news section of this issue.)

PortaAPL

From M Normand Montour

Sir,

In VECTOR Volume 3, No. 1, I contributed an article about PortaAPL on the Mackintosh, where I included a forwarding address for comments. Please could you ask your readers if they could forward enquiries to my new address as given below.

Yours truly,

Normand Montour,

260 Castlefield Ave.,
Toronto, Ontario,
M4R 1G7, Canada.

The cost of APLs

From Mr A N Wiggins

19th February 1987

Sir,

I write this letter for one reason: I am fed up trying to promote the use of APL, when I receive little or no support from the producers of APL interpreters. It seems to me that whilst APL has always had great potential as a language, it has never quite made it.

If I write a program in C, I can compile it into machine code and then give it or sell it to anyone. The price will be a matter between myself and the customer. No one, unless I tell them, would know the original language. In this case I pay a one-off cost and reap the benefits of my labours.

If I write a program in BASIC, I can give it to anyone with the same BASIC interpreter. In terms of (International) business machines, BASIC seems to come whether it is wanted or not.

Our machines are used for accounting functions, so it is not surprising that a copy of a spreadsheet product is bought for each when the machines are purchased; in our case it happens to be SYMPHONY. I can write a program in SYMPHONY for use throughout the department.

So why is there a problem with APL? I can't compile it, so a copy of the interpreter has to be made available for third parties to run my programs. This is where the problem arises.

I use APL*PLUS PC. If I judge that a program is better written in APL*PLUS, I have to provide a copy of Runtime APL with the workspace. That, in itself, does not bother me – but the cost does. Each copy of Runtime APL costs £130, with a minimum purchase of 5; a total of £650.

By comparison, a copy of C can cost as little as £30 or as much as £500. BASIC comes bundled with MS-DOS at a cost of £60. SYMPHONY costs £400 (after discount), but it is usable by anyone after a few hours for simple applications including report formatting. The cost comparison shows that APL*PLUS is effectively a non-starter.

How can this position be resolved to make APL a more attractive proposition for small developers such as myself? One answer is for STSC to price Runtime APL at a minimal amount. The most that should be charged for it is that portion of the cost of MS DOS attributable to its BASIC interpreter. As DOS is a must, the cost relating to that side of the product must be greater than that relating to the BASIC portion. It would not, therefore, seem unreasonable for £20 to be charged for each copy of Runtime APL.

Possibly IBM should automatically bundle their APL with their PCs. This would show that IBM is truly committed to APL, as well as encourage the buyer to try his new "free-bee".

I cannot complain about the cost of my copy of APL*PLUS, nor the cost of the upgrades, as they have proved to be good value for money. But if the banner

"WRITTEN IN APL"

is to appear at the front of more and more programs, then the cost of Runtime APL must come down. This is a way to spread the message and to let the world at large know that APL exists. Keeping it restricted through price will not enhance its use or current reputation; it will only serve to promulgate the myths and mystique with which it is currently surrounded.

I-APL is another way to encourage APL, but surely STSC and the other APL suppliers could nip this project in the bud by making their own products more easily available.

My only interest is in seeing the use of APL grow so that it achieves its rightful place in the computing community. STSC, IBM, and the rest, can do something now . . . or will they eventually sink under a tide of I-APL users?

Yours faithfully,

A N Wiggins,

8 Kidworth Close,
Horley,
Surrey, RH6 8JP.

(Editor: I am sure that if STSC could secure the sales volume achieved by MS BASIC – for whatever reasons – then the economics would enable them to cut prices. If I-APL is the way to stimulate that demand, then maybe STSC and IBM would be wiser not to strangle it at birth.)

British APL Association News

Notice of Annual General Meeting and elections to the Committee

The British APL Association AGM will be held on 5 June 1987 at the Royal Over-Seas League, Park Place, London SW1 starting at 2 pm. The AGM will be followed by a technical meeting as usual.

AGENDA

1 Minutes of 1986 AGM. (These are reproduced below so that they may be taken as read if there are no objections).

2 Officers' Reports: The Chairman will review the highlights of the year, the Secretary and Treasurer will report on the discharge of their offices and the Treasurer will present the accounts. Other officers may take the opportunity if they wish to make a report and members may ask questions relating to their duties of any member of the committee.

3 Nominations of candidates for office 1987/8. These may be sent or given to the Secretary beforehand which would save time. All candidates should be proposed and seconded by members and should provide a brief statement outlining suitability and intentions should they be elected. Candidates should be paid-up members. Standing for election is taken as an undertaking to attend all possible committee meetings.

4 Election of committee: the procedure will be to take the posts in the order below and for each post

  a display the list of candidates;
  b read out each candidate's nomination and statement of suitability and intentions;
  c take a vote by show of hands counted by Chairman and Secretary on each candidate in the order their nominations were received.

5 Any other business. Items will be included only if they have been approved at the Committee meeting to be held on the morning of the AGM. Notify in advance if you want to raise something.

List of posts in the order they are to be filled:

Chairman (who must be a member of the BCS)
Secretary
Treasurer
Journals Officer (Editor and organiser of VECTOR)
Activities Officer (who organises meetings and special events)
Education Officer (who promotes education in APL and APL in education)
Technical Officer (who organises technical vetting and reviews of products, articles and papers)
Publicity Officer (who arranges promotion and coverage of the Association's activities)

Projects Officer (any special projects)
Recruitment Officer (who recruits new members to the Association and advises the committee how to retain old ones)

Committee members are (normally) members of the B.C.S.

Each Officer is encouraged to form a working party of Association members to help with the job, to provide a deputy in case of unavoidable absence and to allow members who might like later to join the committee to see what the work is like.

Anthony Camacho 9th February 1987

Minutes of the A.G.M. held at the Royal Over-Seas League

23rd May 1986

The Chairman, Dick Bowman, reviewed recent progress of the Association. The success of Loughborough had enabled us to launch VECTOR and change the venue for meetings, but last year's bad news was that it was costing us far too much. We would not have been able to carry on if we had not turned VECTOR round: the effort had led to some trouble with regularity of publication but VECTOR Vol 2 No 4 was already printed (to be issued as soon as bound) and VECTOR Vol 3 No 1 would be out in time for APL 86.

Our regular activities were well attended and the new venue well received. There were no special events – APL 86 had taken all the spare effort. We held an Education Day last year which was a stimulating day but has (so far) led to no noteworthy increase in the use of APL in education. We are organising an Education Day before APL 86.

Many of the officers were also on the APL 86 committee and had been working very hard to make it a success. The exhibition has been fully booked, a full programme has been arranged and delegate bookings are buoyant.

If the theme for last year had been consolidation, that for next year must be recruitment. We believe there are at least 2000 people in the UK who should benefit from membership and if a higher proportion were members we would be able to offer more and better services to all. To encourage recruitment, members could earn their subscription by recruiting three new members and the Committee had decided to appoint a recruitment officer.

The Treasurer displayed the accounts. The subscription increase accounted for nearly all the difference in income as recruitment has not been rapid. VECTOR's new production method and increased advertising revenue meant that it was now close to self-financing (not over the whole year shown because that included one issue at the old costs and the transition issue, so the year showed only two issues at the new rate). He held up the membership renewal form being circulated and gave warning that non-payers would not be sent VECTOR. Nobody objected to his proposal not to issue membership cards. Payment will be accepted in dollars or by standing order.

The Secretary held the election of officers for 1986/87. First the offices were listed. Two new offices were added to last year's list. Then nominations for each office were called for in turn – the committee proposing and seconding everyone to save the time that would have been taken by formal proposal and seconding of each candidate – and then, if there was any contention, a vote was taken by show of hands. The following were elected:

Chairman	Dick Bowman	Activities	Roy Tallis
Secretary	Anthony Camacho	Publicity	Gerard Paul-Clark
Treasurer	Mel Chapman	Education	Norman Thomson
Journal	David Preedy	Recruitment	Christine McCree
Technical	David Ziemann	Projects	David Eastwood

It was explained that the Committee wished to be able to initiate projects such as would promote the use of APL, enhance the reputation of the Association, increase its membership or help its finances, without detracting from the attention that members of the Committee could give to their regular work. The Projects Officer could manage any Association projects that were outside the scope of other Committee members or which a Committee member was too busy to handle. Of course the Projects Officer could also propose projects.

24th May 1986:
Anthony Camacho, Hon Sec — Signed: Dick Bowman, Chairman

Secretary's report on Committee meetings

by Anthony Camacho

Since the last AGM there have been seven committee meetings; there will be two more before the 1987 AGM, on 20 March and on the morning of the AGM itself. Below I list the dates and attendances. P means the member was present; A means apologies were sent; R means that the member's resignation had been received since the previous meeting. At the end of each row the numbers separated by the oblique stroke are actual and possible attendances.

	11 Jun	13 Aug	19 Sep	22 Oct	21 Nov	12 Dec	28 Jan	
R J Bowman	P	P	P	P	P	P	P	7/7
A J Camacho	P	P	P	P	P	P	P	7/7
M Chapman	P	P	P	P	A	P	P	6/7
D K Preedy	A	A	P	A	A	P	A	2/7
R Tallis	P	A	P	P	A	R		3/5
P S Goacher							P	1/1
N D Thomson	P	A	P	A	P	A	A	3/7
D M Ziemann	P	P	P	P	A	P	A	5/7
G Paul-Clark	P	R						1/2
B C Leverton			P	P	P	P	A	4/5
D Eastwood	P	P	A	A	P	P	P	5/7
C McCree	P	A	P	P	P	A	P	5/7

The practice that officers send written reports of what has been done and has to be decided to each member of the committee a week in advance is getting to be more common.

Summary of the meetings:

11 June 86:

This was an extra-ordinary meeting for the new officers to take over their responsibilities. The major tasks and budget for each officer were agreed and many minor tasks were actioned. As it was a month before APL 86 many committee members were fully occupied!

13 August 86:

This was an extra-ordinary meeting to discuss the proposed Public Domain Software Library and I-APL. A way of working for the PDSL was agreed for an experimental period of six months. Guidance was given to I-APL on preparation of a case for B.A.A. to support the project. Sales of stock left over from APL 86 were discussed and a journal exchange with other APL groups agreed.

19 September 1986:

The Association's finances were in good shape and its membership too small. There was concern at the amount of volunteer work required to run VECTOR, the Activities and various proposed extra initiatives. A conditional contribution to I-APL was agreed. We heard that APL 86 had made a substantial profit which would take some time to finalise and agreed that we would not use the profit to subsidise regular Association activities.

22 October 1986:

This was an extra-ordinary meeting to discuss recruitment. It was agreed that we have far too small a proportion of the UK's APLers among our members but we don't even know whether our guess that there are between 2000 and 5000 of them is right. We discussed the extent to which our efforts should be directed towards keeping current members, attracting new members from APL users and promoting new use of APL among non-APLers (e.g. in schools). It was pointed out that we are failing to do many simple and obvious things like arranging meetings well in advance and getting them in all the free event listings in magazines and the BCS publications. Most of the discussion was inward looking and restrictive and we discussed a B.A.A. Outstanding Achievement Award (open to members only) and how we could stop non-members benefiting from the Association.

21 November 1986:

This meeting bad-temperedly reviewed the excessively long list of undischarged actions and never got as far as officers' reports. Activities were not getting arranged sufficiently in advance so various people were to be asked for help.

12 December 1986:

An extra-ordinary meeting to discuss a five-year plan. Our relationship with the BCS was discussed and a list made of all five-year plan proposals for expenditure. Roy Tallis resigned.

28 January 1987:

It was agreed we would not raise next year's subscriptions. The five-year plan proposals which had been elaborated were discussed and a subsidised teachers course in APL, a sponsorship scheme for members projects and an outstanding achievement award are to go ahead, together with the Chairman's proposal for better communication between APL groups. We heard that Mine of Information were discontinuing their bookselling activities. Philip Goacher was co-opted as Activities Officer.

I will be pleased to answer questions about the meetings of 20 March and 5 June at the AGM.

Secretary's report on Conscientiousness and the Committee

by Anthony Camacho

The work of the Committee should not be very onerous. It entails attendance at six to nine half-day meetings a year, sending round a report before each and carrying out the actions proper to whatever post is held. The main work is done outside the committee by the officers who have the most responsible jobs and their working groups. The major jobs are VECTOR and organising Activities. There is also a steady workload for the Treasurer, keeping the accounts and membership list up to date and for the Secretary, writing minutes, circulating papers, booking rooms and responding to queries.

When somebody fails to do their duty the whole committee suffers and through them the Association.

It is essential to have reports beforehand, proposals for discussion, a financial statement and a room booked for every meeting. Minutes should come out within a week, letters should be answered and cheques banked promptly and so on.

I am not trying to put you off! Please do stand for the Committee: contested elections are a sign of a thriving Association – but if you are elected please be conscientious about it.

The British APL Association Outstanding Achievement Award

by Anthony Camacho

The British APL Association invites applications for this new annual award. There is no restriction on who may be put forward as a candidate to receive it, but the proposer and seconder must be members of the Association. Any person or group that has achieved some outstanding task with or for APL or performed some outstanding service to the APL community or in the promotion of APL to the non-APL world may be a suitable candidate.

The award for 1987/88 will consist of a Trophy (to be held for a year or until the next time the award is presented) and a cheque for £500. The presentation will be made on a suitable occasion when the Association and recipient will get the maximum publicity.

Each year the nominations are asked for by 30th August. When they are all received the APL Association Committee will appoint a sub-committee, chosen to minimise any associations between judges and candidates, which will decide who should receive it. If, in any year, there are no suitable candidates in the subcommittee's view, then the award may be held over till the following year.

The Association's objectives in promoting this award are to show the world that achievement in APL is rewarded and to get the widest publicity possible for outstanding achievements. We want to encourage efforts on work which might not be commercially viable, and to strengthen and re-inforce the APL community's view about what is best for the future of APL.

Nominations should consist of the name(s) and address(es) of the candidate(s) and two signed and dated statements, one from the proposer and one from the seconder, explaining what the outstanding achievement is and why this year the prize ought to be awarded to this/these candidate(s). The statements should each be not more than a single page of typescript.

Nominations for the 1987/88 award should be sent to the Secretary, British APL Association, 2 Blenheim Road, St Albans, Herts, AL1 4NR not later than 30th August 1987.

£5000 Award for Membership Projects

by David Eastwood

Have you got any pet projects languishing for want of funds? Could you make APL a household name given a bit of time and money? Could you write the definitive APL book? Well the B.A.A. Committee have decided to give you some material assistance.

Following the success of APL86 the B.A.A. has been able to embark on a wider range of activities than in previous years. The B.A.A. committee has launched some new activities, and in addition has decided to support B.A.A. members in projects they would like to undertake themselves. The B.A.A. has allocated the sum of up to £5000 to be awarded in 1987 to the project or projects which will do most to further the aims of the B.A.A., viz:

- Promote the use of APL
- Help people to use APL well
- Contribute to the development of APL and international standards
- Exchange technical information about the language
- Increase the membership of the B.A.A.

If the scheme proves a success it will be repeated in following years, so we are looking for exciting project ideas to get the scheme off the ground. Not all requests for funding will need the full amount so we may decide to sponsor a number of projects, or even none at all! It's all up to the calibre of the submissions. The choice of the project is very much up to you, the members of the B.A.A., but examples might be:

- Writing/publishing a new APL book
- Development of new APL software
- Development of the APL language itself
- Establishing an APL bulletin board

The award can be used to finance time off work, the purchase or rental of equipment, the preparation of material for publication, etc. The awards will be made at the B.A.A. 1987 AGM on June 5th and selection will be made by members of a B.A.A. subcommittee and ratified by the full committee. The rules governing the award are simple:

Principles of the Scheme

1. Projects must be proposed and carried out by paid-up members of the B.A.A..
2. A sum of up to £5000 is available for the project or projects deemed most suitable by the B.A.A. committee.
3. The decision of the B.A.A. committee in choosing suitable projects is final.
4. B.A.A. committee members shall be excluded from judging any project in which they have an involvement.
5. A contract governing the terms and methods of payment will be drawn up between the B.A.A. and the recipients of awards.
6. If the project has some commercial worth, the award will be treated as venture capital funding and the B.A.A. will expect some suitable return on the investment.
7. The awards will be made to the project or projects which, in the opinion of the B.A.A. Committee, are the most original, best further the aims of the B.A.A. and will not be carried out in the absence of this funding.
8. Submissions for project funding should be drawn up to include some budget, timescale and objectives. The progress of the project will be monitored by the B.A.A. committee and in many cases the funding will be released in stages as objectives are met.

Requests for project funding

Proposals for projects should be submitted to the Projects Officer or the Secretary of the B.A.A. Committee by Friday, 15th May 1987 and short-listed candidates will be interviewed by the B.A.A. if necessary.

B.A.A. Projects Officer
David Eastwood
c/o MicroAPL Ltd.,
Unit 1F,
Tideway Industrial Estate,
87 Kirtling St.,
LONDON SW8 5BP

B.A.A. Secretary
Anthony Camacho
2 Blenheim Rd.,
St.Albans,
Herts.
AL1 4NR

Journal Officer's report

by David Preedy

As this is my final issue as editor of VECTOR, a (perhaps misplaced) sense of duty compels me to report to the VECTOR readership on recent performance and future plans of the journal working group.

Over the three years of its existence, VECTOR has made some notable strides. Most importantly it has established an international platform where a blend of articles on more general themes as well as the latest technical developments in APL can be expounded. Already the number of contributors to VECTOR is well over a hundred, although we are always looking to broaden the base of authors still further.

Many of our readers may be unaware of some of the background improvements that have been made. The VECTOR style owes much to the pioneering drive of Robert Bittlestone, its first editor. Since its inception one of the VECTOR group's main targets has been to combine the objectives of maintaining a quality journal with the inevitable limitations of a restricted income. Due to the unstinting efforts of the members of the working group, this objective has been successfully achieved over the last 18 months. We have searched out typesetters, printers and a mailing house who can provide us with the service we require at the most cost-effective rates. At the same time we changed the typeface used so as to improve legibility.

Our internal production methods themselves have changed. Gone are the days when the entire copy had to be entered by the typesetters; almost all the copy is now submitted on diskette – the exceptions being items like the Product Guide where it is more efficient for the typesetters to edit the previous copy. The improvements are seen not only in the costs incurred but also in the task of the proof-readers, the time needed and the final quality of the journal.

We have improved our own timetabling. We would be the first to admit that in the early days, the date on the outside cover was more a wishful hope than a realistic estimate. Deliveries are now normally in the correct month, although even we cannot always allow for the entire copy being held up in a snowdrift for a fortnight!

On the income side, the improved reliability (and the work of our advertising officers) have contributed to a growing list of advertisers. VECTOR is now the prime medium for advertising to the international APL community, be it for new products, consultancy services, conferences or recruitment.

Finally we have started to develop a forward plan of major themes. The coming issues are intended to focus more specifically on the following selected topics:

Vol 4, No 1	July 87	Graphics & user interface
Vol 4, No 2	October 87	APL applications
Vol 4, No 3	January 88	Education

I hope that all readers with an interest in these subjects will take the opportunity to contribute to these special issues.

What then are the major challenges facing the VECTOR working group over the coming months? On the technical side, we still submit all APL code as artwork which is cut and pasted into the type-set copy. This process is not only time-consuming but also liable to introduce errors – the technical editors are unable to simulate all APL environments and so cannot test all the code. We are working on a methodology to enable the complete electronic transfer of articles and code from workspace to type-set copy. As we progress on this front, the results will be published in VECTOR, setting standards for electronic submission to VECTOR.

The second challenge is to shorten lead-times for production. Every so often we find that the passage of time has overtaken the relevance of some of our news material. The previous issue of VECTOR, for instance, contains an advertisement for Mine of Information's APL book service, together with an inserted slip announcing that they have had to discontinue their service.

The final challenge must be to continue striving to generate the highest quality of content. Ultimately the responsibility for that must lie with you, the readers, and I hope that you will take every chance to submit articles, letters and competition entries.

I hope that our readers will judge the history of VECTOR so far as a success. To the extent that they do, the credit lies with all those who have contributed to its content and production. In particular, I would like to acknowledge the conscientious input from all those individuals who have served on the VECTOR working group, and the generous input from those organisations that have supplied us with time, space or equipment.

News from Sustaining members

compiled by Cathy Dargue

"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. Would-be sustaining members should contact the Association Treasurer or any committee member.

This issue we are pleased to welcome yet another company – Peter Cyriax Systems – to the ranks of our sustaining members. It says much for the buoyant activity in the APL market that our original six members (three years ago) have now increased to eleven.

The Committee of the B.A.A. would like to acknowledge the generous financial support of all of our sustaining members. In many cases these organisations also provide manpower and administrative assistance to the Association at their own cost.

APL People Ltd

The A.P.L. Group continues to grow apace. Whilst APL People is expanding its activities as an employment agency, APL Consulting is busy with a number of major contracts in the U.K. and APL Tran Plan has become involved in a car park design study.

Applied Production Logic is the new name under which H Walton Technical Services now operates, following the acquisition of Computerline, the authors of MANTRAC (MANagement by Time Resource And Capacity). This production management package complements the PEFAC system, providing a powerful twin spearhead with which to tackle the manufacturing market.

APL Software has added new mainframe and micro computer packages to its growing portfolio of software. Under a marketing agreement with the Electricity Council, REGGPAK – an APL Regression Analysis Package – is being offered for mainframe (VSAPL, APL2) and APL*PLUS/PC environments. This package has been developed by the Electricity Council to meet their needs for econometric analysis in a way that no other package could. It has been well tested by the area boards in the last three years and the latest release is now to be made available worldwide.

Also included among the latest software releases is IPLS (Integrated PERT Linking System), a PERT-based planning and management system developed over more than five years by British Aerospace because no existing project management system met the engineering needs of large-scale development projects.

APL Software Technology (UK) Ltd.

APL Software Technology is contributing to APL Software's portfolio of APL packages with new releases of both POWERTOOLS/PC and RDS, the Relational Database System.

Release 4 of POWERTOOLS/PC provides some very flexible and fundamental enhancements in the design process for screen-based user interfaces in an already powerful system-building package. This release provides a consistent programming environment for APL*PLUS/PC, Dyalog APL and C, thus giving much greater operating system independence to user applications.

The microcomputer version of RDS, based on Release 4 of POWERTOOLS/PC, is expected to be announced very soon, available for APL*PLUS/PC environments. A Unix-based version is expected to follow.

Cocking and Drury Ltd.

The first U.K trial of the APL*PLUS COMPILER has now been completed. The results clearly illustrate the improvements in performance obtainable through the introduction of compiler technology. A major APL application speeded up by 30% will save £200,000 in computer time, and the reduced machine load means that a costly computer upgrade can be delayed.

APL*PLUS PC is now available on the AMSTRAD PC1512. It runs with an IBM Colour Card compatible soft generated APL character set. The full release 6 product is supported.

APL*PLUS for the DEC VAX range of computers, running under the native DEC VMS operating system, is now being beta-tested in the U.K. The interpreter is based on the APL*PLUS UNIX product and is therefore a full second generation APL interpreter integrated with the VMS operating system.

Cocking and Drury are currently evaluating available PC network software for internal use within the company.

Dyadic Systems Limited

Dyadic is delighted to announce that Dyalog APL has been selected as a "Vendor-Logo" product for the IBM 6150 microcomputer. This means that Dyalog APL for the 6150 will be marketed by IBM itself, and through the IBM 6150 dealer network.

Dyalog APL has so far been announced by IBM in the United Kingdom, France, Germany, Holland, Finland, Norway, Sweden, Denmark, Belgium, Italy, Turkey and Switzerland. IBM's operating companies in the remaining European countries are expected to follow suit shortly; to be followed by IBM's Americas Group and Asia/Pacific Group in the near future.

As an authorised IBM 6150 dealer, Dyadic is pleased to announce a new and more powerful series of IBM 6150 models. The series has two to three times the processing speed of the original models. The floating point performance of up to 1.6 million Whetstone instructions per second is up to 8 times better than before, and disk I/O is three times faster. The system also offers extensive and powerful networking capabilities, and supports 1024 by 1024 graphics displays and windowing.

Dyadic has also released Version 5.0 of Dyalog APL. The latest version contains some major performance enhancements for common scalar operations on large arrays. As a result, applications that make use of large arrays will run up to 3 times faster than before.

Mercia Software Limited

No. 1 news item from Mercia this issue has to be the cracking of the APL*PLUS/PC workspace size limit. With DOS addressing only 640K, APL workspaces couldn't get much bigger than about 487,500 bytes. But now Mercia can supply a memory management unit which lets DOS address 952K – allowing a free workspace of almost 750,000 bytes; it's called the ALL card. It can also help with other PC size problems as it supports LOTUS/INTEL/MICROSOFT EMS – AST EEMS – IBM XMA extended memory specs and a 10 megabyte RAM disk. Prices vary according to your machine and how much RAM you want; for example an ALL card with 1 megabyte for an IBM PC/XT costs £695 – including documentation on how to invoke a large APL workspace (and you will get a 700K battery-backed RAM disk, if your PC had its own 640K to start with).

If you are running out of workspace, and you want (or need) to stay with APL*PLUS/PC under DOS, then Mercia say that this is for you!

On the applications front, old favourites such as STATGRAPHICS, APL*PLUS/PC and EXEC*U*STAT continue to be popular, with Mercia's new STATGRAPHICS customisation service generating interest amongst users who wish to add new statistical routines but don't have the time or APL expertise to do it. Integration into STATGRAPHICS is such that a customised procedure behaves exactly as if it were an original feature of the software – you can't even see the join!

Moving up an order of magnitude, LOGOL should be available by the time this VECTOR hits the streets. Designed by R G Brown for forecasting, inventory management, master scheduling and distribution planning and control, LOGOL is one of the larger and more sophisticated systems written in APL*PLUS/PC. Naturally, with Brown's name behind the system, Mercia are already stimulating plenty of interest for LOGOL.

Peter Cyriax Systems

PCS has been trading successfully as an independent consultant for over four years. As a result of growth and developing areas of special expertise, PCS has now decided to become a Sustaining Member of the British APL Association, and to participate more fully in its activities.

A recent project has brought together such fundamental ideas as fast search structures, database theory, and language design. These have been combined to produce a relational database system of considerable power and flexibility, to which an Application Generator can easily be added; technical Graphics is another natural extension.

This is the basis for further work. PCS will continue to develop their expertise at handling large volumes of data, and expect to announce a commercial database product in due course.

I P Sharp Associates

SWIFT and IP Sharp have entered into a joint venture agreement to market exposure management services to the international banking community. These services will be based on IP Sharp's own or Global Limits package that has already established itself as the market leader. The joint venture will combine IPSA's strength in software development and communications with SWIFT's understanding and experience of the international banking industry. The result will be a larger impact on the banking community.

Reuters have recently bought Securities Clearing International Corporation which, together with IPSA, owned and ran the INSTANT LINK service for the international securities industry. Reuter's multi-million dollar service agreement with IPSA should ensure that the INSTANT LINK becomes the dominant communications medium for the international securities industry.

The city's Big Bang has made borrowing and lending of stocks very attractive to market makers and large fund managers. IPSA's BLEND product is aimed at just this activity and has proven very popular since its recent launch in the UK.The Dow Jones information service has recently been added to the extensive list of databases available through IPSA's INFOSERVICE. The Dow Jones information includes a wide range of textual and numeric information geared to the needs of business.

IPSA's network development project, NET90, is on target and the new communications nodes are now on test within IPSANET. NET90 provides the basis for the next generation of IPSA's communications services and will ensure that IPSA continues to offer state of the art communications to its customers.

One of APL's early strengths was its very convenient applications development environment. However APL has failed to keep up with the development environments now available to other languages. LOGOS is a major product from IPSA designed to provide APL with the tools and controls necessary for large scale APL project development. A new pricing scheme for LOGOS has just been announced based on the number of users of the product. This should make it easier to justify for those companies with relatively small APL development teams.

International APL News

compiled by David Preedy

International conferences

APL87 – Dallas, U.S.A.

With the approach of the APL87 conference in Dallas, the organisers' main announcements require a more rapid distribution than is allowed for by VECTOR's production schedule. I am sure that they will have been mailed out with other B.A.A. publicity in the meantime. The basic information we received from Jim Fiegenschue, provided the dates and venue, which we repeat below:

Venue:	Fairmont Hotel, Dallas, Texas, U.S.A.
Dates:	10-14 May 1987
Contact:	APL87 Registrar, 440 Northlake Shopping Center, Suite 210, Dallas, TX 75238, U.S.A.

APL88 – Sydney, Australia

Consequently our main conference feature looks even further forward, to APL88 in Sydney. Since he prepared the following Call for Papers, Neville Holmes has stayed up to the early hours specially to telephone us with the dramatic news that the Bicentennial cricket test has been scheduled for the week before APL88. I am not sure whether this represents the limitless influence of the APL88 committee keen to attract a large Pommie audience with the prospect of yet another thrashing of the traditional foe, or whether the cricket authorities themselves have recognised the significance of the APL segment of their crowds. In either case this must be an unparalleled opportunity for our friends from Europe and America at last to come to grips with the subtleties of our traditional form of warfare.

APL88 – "APL – Past, Present, Future".

by Neville Holmes

Call for papers

The International APL Conference for 1988 will be held in Sydney, Australia, as an official event of the Australian Bicentenary Year.

The Conference will be held at the University of Sydney, close to the centre of Australia's first city, over Monday to Friday, February 1-5, which is the week after the main Bicentenary Year celebrations.

Because of its historical context, the Conference will have as its theme "APL – Past, Present, Future".

The Organising Committee is therefore looking for papers to be submitted in the following areas :

The Past of APL:

Mathematical notation generally, early days of APL, history of various APL user groups, review of APL machinery and software, review of past APL conferences, development of the ISO APL standard, influence of APL on computing generally.

The Present of APL:

This is the most important and immediate part of the Conference, and will be devoted to applications of APL. The value of APL is best demonstrated by explaining how it has been used in practice to do something which has not been done before, to do something better through use of APL, or simply to do something differently because APL makes it possible.

The Future of APL:

Extensions to the APL notation, improvement of APL interpreters, APL and future workstations, revision of APL and the different approaches to general arrays, directions for the ISO APL standard, the promotion of APL in the computing world generally.

The Organising Committee believes that the future of APL would be best promoted by recognition of present achievements. Too many uses of APL go unnoticed. APL users who have shouted to themselves "Hooray, it works and it is good" should share their joy with others by submitting a paper for APL88.

Submission of Papers

31 May 1987:

Authors should submit a brief description of their intended paper and presentation by the end of May. This will not be refereed but is to help with programme planning. A submission of intent should include a title, a few hundred words outlining what the paper/presentation is about, and mention of any special equipment needed for the presentation. Submissions can be made as detailed below or will be accepted at the APL88 stand at APL87 in Dallas. 'Instructions to Authors' will be sent to submitters as acknowledgement.

31 July 1987:

Final papers must be submitted for refereeing by the end of July. All papers shall be in English, and the body of the paper should not exceed 6000 words. The refereeing process may include requests to the author for modification or explanation.

31 August 1987:

A notice of acceptance will be sent to all authors by the end of August together with any requests for editorial revisions, and details of the presentation time and duration. All accepted papers will be published in the Conference Proceedings.

Modes of Submission:

Electronic submission is to be preferred over hard copy submission, where possible. (This is intended to help the organisers – authors who cannot make electronic submissions will not be discriminated against or disadvantaged except by the various postal organisations.)

Either hard copy or floppy disk submissions may be made to:

APL88 Programme Committee,
P.O. Box 1425,
G.P.O.,
Sydney,
N.S.W. 2001. Australia.

Floppy disk submissions must be labelled (written, not electronic) with the name and address of the author (in case of postal glitches), and include a covering message with author's address and phone number as a file on the floppy disk. Such submissions will be accepted to be run under MS-DOS or PC-DOS (plain text or marked up with DWScript or DisplayWrite).

Direct network submissions may be made to ADR Code NHOLM on the I.P.Sharp network. IBM VNET address SHCHOLM at SYDVM4 is available for IBM employees able to employ this facility.

Direct network submissions must be accompanied by a message explaining what has been sent, how it may be formatted, and how messages may be sent to the author. Submissions may be marked up using DCF (VM/CMS, TSO) or submitted already formatted.

International APL society news

compiled by David Preedy

Swiss APL User Group

We have a couple of amendments to the names and addresses of some of our fellow groups overseas. Dr. J.L. Metzger, the previous acting President of the Swiss APL User Group (SAUG) has written to tell us of his resignation from that role. Their loss is our gain, as he is moving to work with De Beers at Ascot, and he plans to keep in touch with developments in APL on the British scene.

We understand that SAUG is now under the watchful eye of its vice-President Claude Henriod. The address for correspondence is: Swiss APL User Group (SAUG), CH-3000 Bern 1, Switzerland.

APL-Club Germany e.V.

As part of the international APL journal exchange scheme, we have received the latest issue of the German APL-Journal. The 57-page booklet includes a wide variety of notes and articles. Major topics include "Cryptography" – a continuation of an article by Dr Willy Kattwinkel; "STSC's APL compiler: User experience so far" by Jerry Turner; a wealth of product announcements and update information, literature reviews, club information and letters.

More information can be obtained from:

APL-Club Germany e.V.,
z.Hd. Priv.Doz.Dr. C.O. Koehler,
Deutsches Krebsforschungzentrum,
Im Neuenheimer Feld 280,
6900 Heidelberg.

Southwest APL Users Group

Despite the incipient arrival on their doorsteps of the APL87 bandwagon, the Southwest APL Users Group has maintained a busy schedule of meetings and still produces a monthly newsletter which manages to find its way across the Atlantic. Recent meetings have included APL on the Macintosh and a second course in APL by Steve Jaffe. They have also been taking a close look at flat-screen technology. Jim Fiegenschue pointed out that a couple of recent articles in *PC Week* were the direct result of a letter to the editor; he is now encouraging members to write supportive letters to any journal publicising APL describing how the writer himself uses APL. For further details contact: Don Hatfield, 2809 Apple Valley, Garland, Texas 75043.

APL Bay Area Users Group

We also receive regular bulletins from the Bay Area Users Group. The September newsletter explored the area of "Interpolatory, Cubic, Natural Splines" with a paper by Robert Korsan, and a tutorial to be given by Jim Brown on the topic of "APL2 and Artificial Intelligence". October's issue looked at various idioms for handling vector lists of data organized into groups. In November the main topics were a meeting about teaching I.E. and O.R. using APL, and a review of Gene McDonnell's talk on "A Perfect Square Root Routine".

The January newsletter covered the December meeting where Roy Sykes had discussed "Whatever will happen to APL?". The meeting clearly stimulated a keen debate ranging over the scope for further enhancements (those of us who have yet to master nested arrays may be relieved to learn that APL3 is not around the corner), and the bias towards APL as a vehicle for applications rather than user programming (ADRS users neither knew nor cared that it was written in APL and only about 10% of mainframe APL installations used APL for programming). Several reasons were suggested to explain why APL was not used more for programming, and it was suggested that there were too few newcomers being introduced to APL – only one member of the audience had learned APL in the last four years.

New England APL User Group

We also here hear that a New England group has been formed. Despite our initial fears that this was a rival society this side of the Atlantic, it transpires that their address is: New England APL User Group, P.O.Box 2163, Cambridge, MA 02238, U.S.A.

New York SIG-APL

We owe an apology to the New York group whose address has been mis-quoted. It should be: New York SIG-APL, Suite 524, 660 Amsterdam Avenue, New York, NY 10025, U.S.A.

Finally we understand that user groups have been formed in South California and at Princeton. We wish both groups well and hope that they will send us details of what they have been doing.

INFORMATION

- your hidden asset

- Do you have large volumes of data?
- Do you want effective access to the information?
- Would your staff benefit?

If so, we should be talking.
Let us turn your Information Resource into Revenue.
We offer you the skills to achieve this:

Data Audit
Data-Base Design
Project Management
Information Presentation
In-depth Knowledge of Computers
Dedication and Enthusiasm
APL Programming

Interested? then ring Peter Cyriax on:
01 624 7013 (answer phone)
0860 337963 (mobile phone)

Peter Cyriax Systems
213 Goldhurst Terrace, London NW6 3ER

The Education VECTOR

by Norman Thomson

Those who have read the last few VECTORs will, we trust, have got the message that this is the column which tracks the inexorable march of APL into the realms of schools, Further Education, Polytechnics and Universities. On the first of these fronts the news is of the continuing progress of I-APL. Paul Chapman, who is writing the interpreter, is well ahead with the work and demonstration versions will soon be available. On the University front, IBM will soon be giving APL2 to Warwick University, who will make it available to selected individuals for educational and research use. Contact the director of the Computer Centre for more information.

Readers are reminded that I-APL, standing for International APL, is the world-wide free APL, conceived at APL86, to be available as public domain software, that is free for all to copy and distribute without any copyright bar. This is the fireside APL for children to learn at their mother's micro *(Ed: and mothers to learn at their children's micro!)*, but is nonetheless a full-blown ISO standard APL working with APL characters on BBC micros and Sinclair Spectrums (or should it be Spectra?). On standard CP/M and MS/X machines, and also on Apples and Commodores, a transliteration scheme will be necessary so that, for example, the familiar idiom to remove duplicates will appear as:

```
((VīV)=īrV)/V
```

– not too unlike the real thing! Watch this space and keep informed – like a well-known life insurance company, we pay no commission and rely on personal recommendations to broadcast the superiority of our policies!

In July, the British APL Association is sponsoring a 3-day course for teachers and other educationalists at King Alfred's College, Winchester. This will be a concentrated, but we hope also enjoyable, way of involving enthusiastic teachers and others, and thereby spreading the word; more details will be available soon. We want both to talk and to listen, that is to hear the professionals' advice on what is required to promote APL in schools at the fastest possible rate. Of course mathematics is not the only area of teaching to which APL should be applied – statistics, science and social science are all queueing up for attention; nevertheless, mathematics seems a natural first area to tackle in order to hasten the day when the three fundamental skills are seen to be reading, writing and APL – a vision that does not seem to be too outrageous when printed in the columns of this magazine. After all, what is the best way to do maths on a computer?

From school to university the natural progression is from APL to a mix of APL and APL2. The realm of nested arrays, and user-defined operators is one of great untapped worlds of computing possibility, and the arrival of APL2 in the universities is a matter of great promise for the future. There are now at least two University statistics departments which make APL a compulsory course, and this sort of practice must in time produce downward pressure on school mathematics at just the time when I-APL will be there to fill the void. At the other end of the spectrum, we anticipate seeing in due course research in which APL2 will be used to make the computing element both easier and less obtrusive. The simultaneous arrivals of I-APL in schools and APL2 in universities complement each other well.

The third main area of Education is one that could be broadly termed vocational education, i.e. the practical education offered by Technical Colleges, Further Education and the Polytechnics. In this area course content is driven by the demands of employment, and it is not unreasonable that the prime consideration for inclusion in the syllabus should be that people actually use it (whatever "it" is) in the work-place. It is therefore important that APL believers (and it is likely that this term applies to YOU by virtue of the fact that you are reading this journal), who are not directly employed in education, should bring what pressure they can to insist on APL as a prerequisite for employment.

In short, the promise of the immediate future is that Education is the area in which APL will make the fastest progress. The harvest will be truly plenteous – what about signing on as a labourer? (There's room in the workspace for Tories and Alliance too!)

APL Press Review

compiled by David Preedy

For various editorial reasons, the Press Review has been held over for the last couple of issues. The following column reports about 9 months' coverage of APL by the world computing press – time in which as you can see from the length of this column, APL has been hitting the headlines.

In the previous Press Review I looked forward with eager anticipation to the massive coverage of APL86 that we could expect from the UK's computer press. And so as each postal delivery brought another weighty thud onto the doormat, I pounced to pick up the copy so I could include its coverage in time for the VECTOR copy date. Alas, time after time, there was no comment! At the last count I had mustered a grand total of three – and that includes a deliberate plant I inserted in my own letter to *Datalink* (see below).

Computer Weekly (July 17th) managed one entire column inch (38 words no less!), announcing I P Sharp's Logos, which "is said to reproduce many features Unix users enthuse about." The other two mentions were both in *Datalink*, which does continue to be the one weekly newspaper that seems to be conscious of the existence of APL. The on-going correspondence following *Datalink's* unveiling of a major split in the VECTOR working group (see VECTOR, Vol 3, No 1, Page 28) has hopefully been finally put to rest with their publication of a letter from myself (7th. July), and the following week they provided the other coverage of the APL86 Conference under the heading "Double treat for APL programmers." This covered IP Sharp's presentation of their LOGOS product and Cocking & Drury's announcement of release 6 of APL*PLUS/PC with unrestricted variable sizes. Yet again we had a chance to admire Romilly Cocking's smile beaming out to us from *Datalink's* pages; third time this year, and a new photo this time!

Earlier in the year (19th May), *Datalink* had covered a major APL announcement:

> "The APL language is getting a shot in the arm with a tie-up between big North American systems houses, IP Sharp and Scientific Time Sharing Computers."

I can understand both companies' keenness to get immediate national press coverage, but have they thought it might be worthwhile telling the APL community directly via VECTOR?

A couple of weeks later (9th June) and *Datalink* tells us about MicroAPL's release of APL.68000 for the Macintosh, promising future implementations on the Atari ST, with "plans for the Commodore Amiga being held back until the future of Commodore becomes clearer."

In fact so keen do *Datalink* appear to be to publicise APL that our scouts had completely missed two mentions earlier this year. On 24th February they covered Dyadic's announcement of its APL for the IBM 6150/51 or RT PC, and the week before they published a letter from Clark Wiedmann of Haydenville, Massachusetts. This letter ostensibly corrected their comments on the APL compiler which "doubled execution times" (see VECTORs 2.4 and 3.1), but was surely only published to give *Datalink* yet another chance to show us what Romilly really looks like! Wiedmann concludes his letter:

> "It is my contention that the STSC APL*PLUS compiler encourages the production of 'good code' because it removes the interpretive overhead and allows a wider range of programming solutions to be applied. It allows solutions that are clear, forthright, correct, and that would have been considered 'bad code' because of efficiency considerations if a compiler were not available."

It seems that the letters pages are good ground for APL publicity – probably because few of the reporters really know a lot about APL. In *Computer Weekly* (6th March), it was another of our Sustaining Members who went to press. Dave Weatherby, of IP Sharp, was replying to a previous article on the merits of various programming languages for the world of education. In a well-argued letter, Dave points out that:

> "Education involves the understanding of standard concepts and the development of an individual's own ideas through trial and error. The computer is merely a tool in this process and the programs written are likely to be small and used only by their creator for a very short time. Once the ideas have been assimilated the programs will be discarded. . . . Logo is one attempt to address this form of programming. . . . Other languages also address the same area as Logo and I would point to the more advanced versions of APL as having a great deal to offer schools and colleges, especially in the teaching of maths and science at secondary level and above."

Much of the press coverage of APL features new products either in the form of announcements or of product reviews. Fortunately the eager sales managers from most of our Sustaining Members send me copies of their own mentions. For those of you who missed the Dutch review of APL on the QL (see VECTOR Vol 3, No 1), there's been a chance to catch up with a similar review, this time by Glyn Moody in *Practical Computing* (February 1986). The format of most of these reviews is largely predictable, starting as they do from the premise that few readers will have heard of APL, let alone know much about it;

consequently there is the obligatory line pointing out how APL can add two numbers in 'desk-calculator' mode, and so on. The variations tend to come in the final summary. In this case the article concludes:

> "There is no doubt that freed from the shackles of an impenetrable notation, APL emerges as a very powerful and usable language. . . . . APL represents one of the first products simultaneously to use something like the real power of the QL as well as offering the serious user or professional the possibility of advanced programming on a home micro."

Byte (March 1986 issue) also had a long product review – this time of Pocket APL and spread over some seven pages. Obviously Byte readers are regarded as APLiterate, since they are denied the chance to see how 1 and 1 can be made to make 2. Instead they were given two separate comparisons of how to add up a list of integers in APL and in BASIC. This review made several reasonably sophisticated comparisons between Pocket APL and BASICA on the IBM PC. Out of nine benchmarks, Pocket APL won on seven comparisons, lost barely on one test, and was comprehensively beaten on the 'calculations' benchmark, where as the reviewer pointed out, "there were no parallel operations". But Eric Johnson made some other important comparisons:

> " . . . show some of the advantages of using APL, in terms of time, accuracy and the generality of the solutions. But APL also has a dark side. While the BASIC program and variables . . . occupied only 132 bytes of memory, Pocket APL hogged nearly 20K bytes to do the same job."

On the subject of parallel processing, *Computing's* "Over the Horizon" series looked at the area of Vectorising Compilers (19th June). This article was interesting for more than the claim that:

> "the future of VECTOR processing is secure" (my capitals)

which is a relief to all members of this journal's working group! However Tony Durham's article has several more direct references to APL, as he compares how an APL programmer might approach the problems of parallel processing, compared to, say, a Fortranner.

> "To an APL programmer it might seem perverse that Fortran programmers are carefully specifying the sequence of operations which, as the smart compiler will discover, could actually be done in any order.
>
> "But habits of thought are strong. The APL programmer would be uncomfortable thinking explicitly about sequence. The Fortran programmer would be uncomfortable thinking explicitly about parallelism. The world of supercomputing is full of Fortran programmers."

Later, Tony Durham looks towards special languages for supercomputing.

> "There is still no sign of a serious move to vector or array languages, which match the Cray architecture better than Fortran or C. APL is a non-starter for supercomputing because it is interpreted, not compiled. It also uses unusual symbols and needs a special terminal.

> "But Wallach *(Ed: Steve Wallach, Technical vice-president at Convex, one of the firms developing cost-effective alternatives to the Cray machines)* who once designed an APL machine, says it would be interesting if someone produced a good APL compiler. He and his colleagues actually used APL to describe some of the algorithms they used in their own Fortran compiler."

Back to *Byte*, whose May issue again featured APL, albeit less flatteringly than in the Pocket APL review. The letters page, under the caption "Choose Your Language" argues for having a range of languages in one's repertoire. The illustration given is one of those popular lists of undesirable combinations:

> "A fatal flaw amongst programmers is the desire to use one language for everything. Do you drive nails with a pair of pliers? Cut wood with a butter knife? Write numeric-intensive applications in C, screen editors in COBOL, or system code in APL?"

Later in the same issue, Kent Smith, reviewing Easy C, makes a similarly unfavourable comment:

> "Because we have had bad experiences with other powerful but 'compact' languages (most notably APL), we were also less than thrilled by our first exposure to C."

It's always interesting to read the views of a genuine user, rather than the hype of the salesmen or the padding of the professional reviewer who rarely uses the product reviewed in anger. I was pleased therefore to see *Computer News* (29th May), where Helen Sturridge looked at the link between Information Centres and PCs. The user chosen was the Hoechst company which introduced Apple IIs in 1981:

> "It was the development backlog that sparked an interest in the new machines at Hoechst UK. But the interest was coupled with a desire to offer mainframe personal computing through APL. . . . . Mainframe APL users are divided between those who want sophisticated graphics and those who want access to corporate databases. Some do it through PCs, some through terminals."

In September, APL finally made it into the hallowed pages of *The Times* (Computer Horizons, 9th September). Chris Naylor was reporting on the mass of languages available. APL merited 19 words:

> "APL simply stands for A Programming Language and contains particularly powerful commands for doing such things as matrix operations."

The descriptions of other languages were similarly terse, with more space devoted to the origins of their names than to their characteristics. The description of Ada devotes 16 lines to Lady Lovelace and Babbage's Engine and only 5 to the US Department of Defense.

Later in September, *Computer Weekly* took a look at Parallel Processing with an article by Dr Chris Jesshope from Southampton University. After looking at the communications bottleneck – "typically the processor-memory interface, the bane of the von Neumann architecture" – he turns to software and Occam.

> "Describing structure parallelism in Occam, although possible, is rather awkward. Notations and techniques that were pioneered by Iverson in APL are far more suitable for expressing this form of parallelism.
>
> "In fact Fortran, the language that refuses to die, is about to be blessed in the current Ansi standards deliberations, with constructs which support structural parallelism, as applied to arrays. Such facilities may be added to Ada, although not supported in the language standard."

In October, *PC Week* reviewed the most widely used computer languages, concluding that "Cobol and Fortran have stood the test of time" (8th October). APL was included in the glossary of languages:

> "APL: A Programming Language, designed for solving mathematical problems, noted for its matrix-generation capabilities."

However, the body of the article contained no reference to APL, even though Pascal, C, Modula 2, Prolog, Lisp, Forth, Basic, Algol and Ada were mentioned.

We were not the only people to notice this omission. On 26th November, *PC Week* published a two page response by Peter Branson. Peter picks up on some of the myths propagated in the previous articles – its unsuitability for business applications, dependence on expensive memory, and the dreaded 'Greek symbols':

> "The fact that it uses funny Greek symbols for many of its operators is true, but if you don't like them use a keyword version which reads as easily as BASIC."

(Editor: In fact it only uses a handful of Greek symbols, and they are not particularly humorous as Hellenic hieroglyphs go!)

Peter Branson then proceeds to show how the Simpson numerical integration code – used in the previous series – can be written in the KEY*PL dialect. Peter also gave some publicity to the I-APL project, although he mistakenly gave credit for its inception to the B.A.A. It is, of course, an independent venture.

Still in November, *EXE*'s News Editor, Nick Roach, gave his "first impression of the APL community", under a bold banner heading:

> "Not Just For Nutters".

Having spent some time at MicroAPL, he commented:

> "A glance at the APL world may give programmers and DP managers reason to look more seriously at APL for mainstream applications."

He then proceeds to survey the history, the hardware, the language and a variety of applications of APL. The hardware coverage may have been, perhaps understandably, somewhat biased towards the 68000-based machines:

> "IBM PC machines have hardly had a look in to the APL scene"

but the applications sections mentioned several good examples of APL in business use. Roach concludes:

> "Word is out from Xephon that IBM's given APL a low priority in preference for a thing called AS. If that's the case, the continuance of APL could well be due solely to the micro-based suppliers and consultancies. In the U.K., it's precisely these people who've been driving APL. I'm sure they'll continue."

And just to make the point he listed the principal APL UK suppliers – a list remarkably similar to our own list of Sustaining Members!

Despite his writing a fairly favourable article, Nick Roach found out in *EXE's* January 87 issue that the APL community will always rise to the language's defence in order to "respond to, and hopefully clarify, a number of statements." In this case the knight in shining armour on the letters page was David Ziemann. Among the points he challenged were the definition of APL as a 'mathematical' language, the belief that it is inevitably heavy on machine time, the 'abnormality' of the right-to-left evaluation, and the problems of reading and understanding someone else's code.

Back to November and our old friends *Datalink*. Their front page was emblazoned with the news that "APL is fast falling out of favour", covering the Xephon survey of Information Centres. APL is described as "cumbersome, difficult to use and inefficient", and the report describes the emergence of AS which "IBM has consistently pushed as the strategic Information Centre and decision support product". By the following week things had clearly deteriorated still further because by 17th November, Kenny McIver was able to tell *Datalink* readers that "APL, once the big hope for application developers, has been all but abandoned". McIver gives fuller coverage to the Xephon report and includes an interview (and the ubiquitous photograph) of guess-who – Romilly Cocking! Romilly points out that "APL should never have been used as an end-user language" and believes that IBM has now changed its plans for APL:

> "IBM's key information centre product, InfoCenter/1, is implemented in APL and will continue to be strategic but not in the way that IBM originally envisaged."

However, McIver does point out an atmosphere of optimism in the APL community:

> "The general impression in the APL community is that the language is on the edge of an explosion, as big as the boost it received with IBM's adoption of APL for the information centres in the 1970s."

Romilly confirms this, identifying the two factors fuelling such growth as the relatively recent availability of APL on the PC and the introduction of APL compilers.

Yet again, however, the adverse publicity is not accepted silently by the world at large. Two weeks later, on turning the the *Datalink* letter page, we hear that "APL is alive and kicking". On this occasion the correspondent is Peter Donnelly of Dyadic Systems. Peter points out that the Xephon report was by definition restricted to mainframe uses where the user "must choose between writing an entire application in APL or not using it at all". Peter concludes with a strident defence of end-user APL:

> "Finally, I must disagree with the view that APL is 'unfriendly' and unsuitable for end-user computing. It is true that it is not everyone's cup of tea; but neither is Basic. APL is superb for quantitative analysis and is a strong favourite among actuaries, investment analysts, statisticians, planners and other numerate professionals. The Big Bang has brought about a much more analytical approach in the City, and a growing number of analysts look to APL on super-micros for their computing needs."

Meanwhile *Data Processing* (November 1986) published three pages on the "Uses and limitations of APL" written by Anthony Camacho. Unfortunately Anthony's article encountered several attacks of the printer's devil, who kindly shuffled most of the paragraphs on the first page. It appeared that the history of APL jumped without any significant event between the first implementation on a computer in 1964 to VIZ::APL in 1982. There was an interesting section on the Bang and Olufsen system, as described at APL86, and an exposition on the benefits of prototyping and some of the main reasons why that approach has yet to be fully exploited. However, I found the article somewhat disjointed with rapid jumps from one theme to another, as if a somewhat cack-handed editor had been at work. So we found a useful discussion of the attributes of an APL programmer and the benefits of thinking with parallel processes went under the sub-heading:

> "APL is not COBOL"

which was not only self-evident, but also irrelevant. No doubt David Ziemann will also write a letter reproaching the author for describing APL as the "mathematically-based programming language"!

Still in November, obviously a bumper month for APL publicists, *Computer Weekly* (November 13th) has a Background Briefing on Fourth Generation Languages. One sub-category are those "conventional third-generation programming languages which, in the view of their advocates, are substantially more advanced than the others" in respect of the fourth-generation characteristics. RPG II and APL, which "has a very high level of functionality per source program statement" are the only two languages specifically cited in this sub-category.

November and December were good months for publicity in America as well, and I am grateful to our friends in the Bay Area and South West User Groups who spotted the following articles. In *IEEE Spectrum*, Paul Wallich and Glenn Zorpette discussed "Whatever happened to APL?" *(Editor: It's noticeable that the Americans seem to think that everybody has heard of APL; the British assume that nobody has!)* They quote a Hacker's proverb:

> "It took God seven days to create the world, but an APL programmer could do it in one line."

The article tries to explain why APL has fallen from favour, particularly in the academic community, even though "it is used for banking, brokerage, image processing, database operations and any number of other applications." Having cited a number of case-studies from business and science – mainly drawn from the New York SIGAPL's successful seminars "APL: a tool of thought" – the authors point out:

> "The CRT terminals that spread rapidly through the computing world during that period (about 1980) were for the most part unable to display APL characters. (Allen) Rose attributed the decline of the language in significant part to its non-standard character set and the insistence of Falkoff and Iverson on retaining it."

They then describe APL's continuing popular status in the important area of the information centre, and point out the growing popularity of APL on personal computers.

In *PC Week* (2nd December) there were three linked articles by Winn L. Rosch. The first describes how "APL Developer Recalls the Language's Birth" and reports a lengthy interview with Ken Iverson; the second comments on how "Business Users Flock to 'Oddball' Language"; the final section shows how "APL Language Helps Developers and Analysts in the Design of Software".

Only about a third of the interview with Ken Iverson concentrates on the development history of APL. The rest summarises the key concepts of APL as an extended mathematical notation:

> "APL is different from mathematics because mathematics is incomplete"

and then highlights some of the strengths of implemented APL over conventional computer languages such as BASIC and Fortran. Finally the article reports Ken Iverson's explanations of why APL has remained a minority language:

> "The reluctance to adopt something new is not specific to APL, even if you're just looking at notation. I like to think that APL is in good company. Look at something that was even a more important step forward than APL – the decimal system. Compare it to the Roman numeral system that it replaced. You'd think something as efficient as decimal notation would be adopted in not more than a week. But it took centuries – and even longer in commercial data processing."

The interview concludes:

> "The greatest strength of APL is that it makes it possible for people who know a particular area of application to use a computer effectively. If someone has been doing an inventory for a number of years, he knows more about that than you or I or any programmer. In many cases you will find that the people who do the most interesting work are those who learned APL themselves and simply use it."

The second of *PC Week*'s articles reports the views of three executives at STSC – Pat Buteux, Clark Wiedmann and Jerry Turner. The article gave a bullish outlook on the use of APL, explaining why APL actually gets used for business applications:

"The problems that the typical APL user must solve, Pat Buteux explained, are too large and complex for off-the-shelf data-analysis software such as spreadsheets. 'They are people who have large amounts of data to go through – one guy told me he had to solve 50,000 simultaneous equations. They are not necessarily mathematicians. They want to use maths to solve their own day-to-day problems.' "

Clark Wiedmann points out:

"something of the order of 3 to 5 percent of programming is being done in APL .... What makes APL special is that it has a devoted following of people who are its advocates. Few languages can boast regularly held conferences concerned particularly with them. APL is a unique phenomenon."

And Jerry Turner's conclusion is that:

"APL is fun to program in."

The smallest of the *PC Week* articles describes APL's use as a prototyping tool at CBS Magazines and by Qualitas Inc. Chris Oakleaf of CBS Magazines summarises:

"When prototyping in APL, you sit down with man and machine and immediately begin to implement the idea. You've got the guy there, and he can answer your questions right away. Because you are dealing with a prototyping language, you can quickly change anything."

More recently, and moving back to this side of the Atlantic, *Computer News* covered the announcement by Mercia Software that it has cracked the problem of limited APL workspace on PCs. Gareth Brentnall said that he believes "this breakthrough will breathe new life into APL generally" and an enthusiastic response was also forthcoming from an APL Association spokesman – Anthony Camacho, who predicted new growth in the use of APL on PCs, but doesn't believe there will "necessarily be any trend away from mainframe APL use".

The 'big three' U.K. weeklies all mentioned APL in the last week of January. In *Computer Weekly,* Michael Powell reported that Cobol remained very much the dominant commercial language:

"But there are other languages, such as LISP and APL, which have been identified as vehicles for the future. The latter was tipped as important in the 1970s but never quite made it. According to David Weatherby of I.P. Sharp, this was because of the cost of machines and memory, which is no longer so important."

On January 29th, the *Computing* letters page included a strong defence of Cobol by Barrie Thompson, from Sunderland Polytechnic. He was reporting on the experiences of his data processing students during their sandwich year in industry.

> "42 reported the use of Cobol . . . 22 reported some use of 4GLs (as identified by James Martin in his text Fourth Generation Languages and hence included APL, Filetab and Dbase as well as languages such as Natural and Mantis); and two reported a minor use of Pascal."

In *Datalink* (26th January), Frank Brett went "Blue all over", reviewing IBM's mainframe software, with more reports of APL's decline within IBM.

> "Despite the growing use of personal computers, end-user computing on the mainframe is still very much alive and well. The tools being used are changing, however. The APL and APL-based products which used to be popular appear to be falling out of favour. Gaining in popularity is a relatively new product, Application System (AS) which can be run under VM or MVS (TSO)."

Interestingly, the following week (2nd February), *Datalink* reported that "IBM unveils better Fortran, APL", announcing that a release date for the APL2 Vector Capability had been set for Spring 1988. "IBM", we read, "is promising to enhance APL2 . . . so that it will be directly supported by the 3090 Vector Facility", which will allow vector processing to speed up large number-crunching exercises.

The final wave of publicity in time for this column was led, again, by *Datalink* (23rd February) under the banner:

> "APL altruists aim to assist schoolchildren".

The article reported the work of the I-APL project and its aims to provide a public-domain interpreter, targetted largely towards schools. Similar reports appeared in the March issues of *Practical Computing* and *Personal Computer World*. The former gave a straightforward announcement, but *PCW's* coverage was presented under the more colourful headline "Religious instruction". It started:

> "A Programming Language, APL, is a religion which even Sinclair Spectrum users can now have. Well,soon. A group of religicos called I-APL, the Free International APL group, has decided to write and issue an interpreter."

and finished:

> "I'm not getting involved in religion. It's a programming language, and mathematicians like it, and it needs special characters, and how they're going to make the same version run on Spectrum and IBM machines, I'm blessed if I can guess. But I'm not getting involved."

No mention, unfortunately, of the targetting towards schools, nor that the interpreter will be free. Still, I suppose some publicity is better than none!

Moving to less serious topics it was reported in *PC Business World* that apparently "Tandy is facing a summer rebellion among employees having just decreed that beards and the Miami Vice look will not be tolerated in its special Computer Centres. As one manager put it, 'shave and wear a dark suit, or be fired'." Well I hope such extreme rulings don't cross the Atlantic too quickly, or the B.A.A. will be low on committee members!

Our reporters were also really excited to come across an advertisement for an APL not included in our extensive product list, advertised in, of all places, *Farmer's Weekly* under the banner heading:

"APL 300 – gives better steering, whichever way you turn"

This is undoubtedly going to challenge the ISO standards committee since it suggests that in operation there may be a different lock to the right and left! Fortunately it's all concerned with tractor axles. However pride of place this issue must go to that renowned and much-read journal *PSLG – Public Service & Local Government* (February 1986). Where else could you read within one issue about such diverse topics as the merits of mulching, asbestos removal, care of bowling greens, vandalism at schools, Avon sending its rubbish by rail and a 'new' computer language. In a mere page, Henry Law describes the history of APL from its invention by Ken Iverson, through to its advantages in modern-day computing:

> "What is likely to boost APL is the rising cost of programming time and the tumbling cost of computing power; after nearly 25 years in relative obscurity, APL looks set to catch on. . . . . An IBM commentator has predicted that within five years, the use of APL could increase from its present 5 per cent of all computer applications to a dominating 50 per cent."

It must leave commentators on the computing world with a difficult decision. Do they believe the Xephon report or the more prestigious *PSLG*?

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. 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
Volume 2	10	14.00	22.00

Please send sterling cheques or money orders (payable to The British APL Association) to the Treasurer:

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

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

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.

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 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
- ★ APL Training Courses
- ★ Other services
- ★ Vendor addresses

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

Note: 'poa' indicates 'price on application'

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

Steve Lyus
Metapraxis Ltd.,
Hanover House
Coombe Road, Kingston
KT2 7AH

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES £	DETAILS
Analogic	The APL Machine	$60,000+	AP500 array processor, 4 Mb data memory, 80 Mb disk drive.
Dyadic	Dyalog APL Coprocessor	3,500+	32-bit coprocessor board for IBM PC. NS32000 cpu with FPP, up to 4Mb RAM, 16Mb virtual memory. Software includes Unix V.2, Dyalog APL, graphics support, DOS interface. Provides multi-user Unix/DOS environment.
	IBM 6150	15,000+	Multi-user Dyalog APL system with Fast 32-bit RISC processor, FPP, up to 8Mb RAM, 210Mb Disk, 16 users. Interface to SQL, graphics and APL support for standard IBM peripherals.
	Altos 3068	25,000+	Multi-user Dyalog APL system with MC68020 cpu & MC68881 FPP. Also features a LAN which supports IBM PCs as Dyalog APL terminals.
	Sun 3	15,000+	Multi-user Dyalog APL systems which can be configured as a network of workstations and or a traditional time-sharing cpu. With its 25MHZ 68020 cpu, the Sun 3/200 is the fastest APL microcomputer on the market.
Gen. Software	Myriade	poa	TI computer + APL & APL operating system
Inner Product	IBM PC	2,000 –6,000	IBM PCs supplied for turnkey applications
M.B.T.	MBT Series 10	poa	UNIX/68010 based multi-user APL system
	TORCH	poa	68000/Z80 multiprocessor
MetaTechnics	—	poa	Details on application – IBM PC compatible
MicroAPL	Aurora	23,500	Multi-user APL computer using 68020 CPU. Std. configuration 2Mb RAM, 16 RS232 ports, 68 Mb hard disc, 720K diskette
	SPECTRUM	11,000 –15,000	Expandable multi-user APL computer using Motorola 68000. Std. configuration 1 Mb RAM, 12/36 Mb disc, 12 ports.
	STRIDE 440	8,500	Multi-user APL computer, 1 Mb RAM, 12/18 Mb disc.
	Atari 1040ST	799 –999	1 Mb Mono/Colour System, includes 1 Mb disc drive & mains transformer built into Console.

APL TIMESHARING SERVICES

COMPANY	PRODUCT	PRICES £	DETAILS
Boeing	Mainstream APL	poa	Enhanced IBM VS APL (CMS)
Mercia	APL*PLUS	poa	STSC's Mainframe Service – MAILBOX etc.
I.P. Sharp	SHARP APL	poa	International Network application systems and public databases.
Uniware	APL*PLUS	call	STSC's mainframe service

APL INTERPRETERS

COMPANY	PRODUCT	PRICES £	DETAILS
APL Software	Dyalog APL	1,000-8,000	See Dyadic Systems entry
Cocking/Drury	APL*PLUS/PC Rel 6	475	STSC's full featured APL for IBM PC, PC/AT and compatibles
	Upgrade 5 to 6	120	Extension from rel 5 which incorporates 64K object support.
	Upgrade 2,3,4 to 6	220	Extension upgrades to release 6.
	Run-time	poa	Closed version of APL*PLUS/PC which prevents user exposure to APL.
	APL*PLUS UNIX	poa	STSC's 2nd generation APL for IBM PC/AT, DEC, AT&T and other Unix computers.
Dyadic	Dyalog APL	795 -10,000	2nd gen. APL for UNIX systems, e.g. IBM 6150, Sun, Vax, NCR, HP9000, AT&T, Altos, Apollo, Whitechapel, Sperry, etc.
Gen. Software	APL*MYRIADE	poa	Runs on Texas Instruments TI990 range.
IBM UK Product Sales	IBM PC APL	poa	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 Rel 6	450	STSC's full-feature APL for IBM PC, and compatibles. No 64K object size limit.
	Upgrades 2,3 & 4-6	225	
	Upgrades 5 to 6	130	
	APL*PLUS/UNIX	poa	Interpreter for UNIX systems: WICAT, CADMUS, CALLAN, FORTUNE 32:16, HP, 9000/500, OLIVETTI 3B2, SUN etc.
MetaTechnics	APL*PLUS Rel 6	475	Discount on quantity.
MicroAPL	APL.68000	1,000+	Full implementation with component files, error trapping etc. for SPECTRUM, SAGE & other MC68000-based computers.
	QL/APL (keyword)	87	Full keyword APL for QL with many extra features.
	QL/APL (APL chars)	87	VSAPL compatible APL for QL with many extra features.
	APL.68000 for Apple Macintosh	257	
	APL.68000 for Commodore Amiga	200	
	APL.68000 for Atari ST	170	
	APL*PLUS/PC - REL 6	450	
Portable	PortAPL	$195	IBM PC Software
		$275	Mackintosh
		$2,995	DEC VAX
I.P. Sharp	Sharp APL/PCX	2,575	For IBM XT/AT
		1,000+	For IBM mainframes
	Sharp APL/PC	325	For IBM PC or PC/XT
Uniware	APL*PLUS/PC	495	STSC's full feature APL for
	Release 6	call	IBM PC/XT/AT, Compaq, Olivetti
	Release 5 update	call	Extension upgrade from release 5
	Release 4 update	call	Extension upgrade from release 4
	Release 3 update	call	Extension upgrade from release 3
	Run-Time	call	Closed version of APL*PLUS/PC which prevents user exposure to APL
	APL*PLUS/UNX	call	STSC's full feature APL for UNIX based computers.
	PortaAPL	280	PORTABLE SOFTWARE's APL for APPLE MACINTOSH.
		2,995	PORTABLE SOFTWARE's APL for the DEC VAX.

APL VISUAL DISPLAY UNITS

COMPANY	PRODUCT	PRICES £	DETAILS
Dyadic	Lynwood j300	1,560	Monochrome ANSI 3.64 APL vdu, 15-inch high quality screen, Tek graphics, local macro keys.
	Lynwood j500	2,295	Colour ANSI 3.64 APL vdu, 15-inch high quality screen, Tek graphics, local macro keys.
	IBM 3163	791	Low-cost Monochrome APL vdu. Supports downloaded Dyalog APL font.
	IBM 3164	1,093	Low-cost Colour APL vdu. Supports downloaded Dyalog APL font.
Farnell	Tandberg TDV 2221	995	Ergonomic design APL terminal, 50-19200 baud, 15" anti-reflex screen, low profile keyboard
	Tandberg TDV 2271	1,195	Combined APL/ANSI ergonomic terminal as above.
Gen. Software	Mellordata Elite 3045A	400	Second-hand
M.B.T.	various		Contact MBT for details
Meta Technics	IBM EGA compatible	299	Emulates EGA & Hercules, Half Card
MicroAPL	Insight VDT-1	795	Inexpensive APL VDU
	Insight GDT-1	1,450	With monochrome graphics
	Concept 201	1,295	APL VDU with 8 page memory
	Concept 201G	1,650	Graphics VDU
Shandell	HDS2010	1,215	ANSI3.64 DEC VT52/100/220 compatible. 15" tilt/swivel screen, low profile keyboard 8 page memory, windows, viewports, 80/132 columns, full overstrike, 2 or 3 comms, ports, 55 PF keys, NVM storage.
	HDS2010G/GX	1495+	As above plus Tektronix 4014, Retrographics VT640/D0640 and Visual 500 compatible. 1024 x 390 or 1024 x 780 resolution.
Tektronix	4114B	13,500+	19" D.V.S.T.:Graphics: 3120 x 4096 displayable; Intelligent: up to 800K memory; APL keyboard (option 4E)
	4125	21,550+	19" 2D colour graphics; Workstation (1280 x 1024);Intelligent: up to 800K memory; APL keyboard (mod AP)
	4128	26,822+	As 4125 plus 3D wireframe

APL PRINTERS

COMPANY	PRODUCT	PRICES £	DETAILS
Datatrade	Datasouth DS180+	1,295	180 cps matrix printer with 4K buffer, 9 x 7 dot matrix and APL option.
	Datasouth DS220	1,695	Letter quality; graphics capability, APL option (both available with IBM Twinex or Coax interface).
Dyadic	IBM 4201 Proprinter	poa	100, 200, 40(nlq) cps, matrix printer, with graphics. Supports downloaded Dyalog APL font.
	Toshiba P351	poa	24 pin high-quality matrix printer 100 cps letter quality, 192 cps draft.
Inner Product	Epson FX80	500	Soft char. set, 160 cps, 80 column
	Anadex 9620	1,150	200 cps., 132 col., tractor feed
	Siemens PT88	620	180 cps., 80 col., silent
	TGC Starwriter	1,180	40 cps., letter quality
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+	1,295	See Datatrade entry
	Philips GP300	1,924	Matrix printer with letter & draft quality and APL.
	Qume Letterpro20	549	APL/ASCII Daisy-wheel printer

APL PACKAGES

COMPANY	PRODUCT		PRICES £	DETAILS
APL◇385	FSM 385	PC:	50	Screen development
	DRAW 385			Screen design
	DB 385	Mainframe:	125	Relational W.S.
	GEN 385			Utilities
APL Software Ltd	*Mainframe*			
	AFM/AP		11,035	Interprocess Software for VM/CMS & MVS/TSO.
	– Keyed Access		2,650	Component File Management System (VSAPL/APL2)
	– Interactive Link		1,325	
	– Mail Exchange		2,650	
	CALL/AP		4,030	Non-APL program execution (VSAPL/APL2)
	APLPRINT		2,205	Output to high speed line printer or 328x devices (VSAPL/APL2)
	ENHANCED FORMAT		2,205	Extends Format operator to full "Quad-FMT" status (VSAPL/APL2)
	ISP		750	Input and Output Stack Processors for manipulating terminal I/O
	OSP		2,205	with facilities for Error Trapping (VSAPL)
	DISPLAY CAPTURE		poa	Allows terminal output to be collected and held for retrieval by an APL function (APL2)
	UCF		poa	User Communication Facility for data transfer between users (APL2)
	RDS		poa	Relation Data Base System
	PANEL		poa	Fullscreen management system
	PFS		poa	Program File System – APL Systems development aid
	IPLS		poa	Project Management System
	REGGPAK		poa	Regression Analysis Package
	Microcomputer			
	POWERTOOLS		295	Assembler written replacement function for commonly used CPU-consuming APL functions, includes a Forms Processor.
	Microcomputer			
	REGGPAK		poa	Regression Analysis Package
	RDS		990	Relational Database System
Beta-plan	BETA-FONT		poa	Multiple font PC character generator. Dealers required for non-Scandinavian countries.
Boeing	TABAPL		poa	Hierarchical Planning System
Butel	Merlin		5,000	Mainframe APL spreadsheet runs under VM/CMS, TSO, VSPC
	Merlin/PC		poa	Version for APL*PLUS/PC
Cocking/Drury	*For VSAPL*			
	STSC's SHAREFILE & enhancements to VSAPL		30,000	Component files, quad-functions & nested arrays for IBM VSAPL under VM/CMS & MVS/TSO
	SHAREFILE only		15,000	
	ENHANCEMENTS only		17,000	
	COMPILER		30,000	First APL compiler. Available with APL*PLUS enhancements and Sharefile under VM/CMS & MVS/TSO
	FILEPRINT		8,000	Print APL component files
	FILESORT		8,000	Sort APL component files
	FILECONVERT		8,000	Convert non-APL files to APL files
	FILEMANAGER		8,000	Extends APL primitives to database management
	TOOLS + UTILITIES		8,000	APL Software development tools
	DATAPORT		poa	Powerful Information Centre spreadsheet incorporating data exchange between APL and FOCUS, IFPS, SAS, APL/DI, ADRSII, LOTUS123, VISICALC, MULTIPLAN, DIF files
	For APL2			
	SHAREFILE/AP		15,000	STSC's sharefile for APL2
	FMT		2,000	Full featured FMT for APL2
	WSDOC		5,000	
	FILEMANAGER		8,000	

	Microcomputer			
	STATGRAPHICS Rel 2		545	Powerful Statistics and graphics on IBM PC's, PC/AT's and compatibles
	Release 2 update		165	Update from release 1 to release 2
	APL*PLUS PC Tools			
		VOL 1	325	Incl. 327x IRMA support, RAM disk, full screen data entry, menu input, report generation, games.
		VOL 2	125	Incl.file documentor, screen editor, exception handler.
	APL*PLUS PC Fin & Stat. Library		350	Financial & statistical routines
	SPREADSHEET MANAGER		195	APL-based spreadsheet for APL*PLUS/PC. Cell arithmetic; transfers to ASCII, LOTUS
	APL Debugger		95	Debugging tool for APL*PLUS PC
	UNITAB		250	Spreadsheet for APL*PLUS PC
E&S	PROTOPAK			Packages for prototyping management information systems – PC & mainframe
	consisting of:			
	RMS	Modules		Relational databases.
	AMS		250+	Multi-dimensional arrays
	RAMS			Combined RMS & AMS.
	BMS			Dynamic financial modelling & forecasting
	FMS			Full-screen handler for APL*PLUS/PC. (AP 124-based)
	CMS			Communications package.
	SOS		poa	Scheduled ordering and stock control.
Gen. Software	PROPS		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		3,500 –10,000	Accounting package & 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		3,500	EEC monetary and agrimonetary analysis.
M.B.T.	RHOMBUS		poa	Integrated Office System
	HASLEMERE		poa	Hotel Accounting System
Mercia	STATGRAPHICS 2		535	Integrated stat. graphic system for PCs.
	Upgrade to Release 2		175	
	EXEC*U*STAT		395	Easy to use Statistics for management.
	APL*PLUS tools			
		VOL 1	225	IBM PC Utilities:IRMA3270 comms, full screen, RAM Disk report generator
		VOL 2	125	File documentation, screen editing. Exception handling.
	FINANCIAL AND STATISTICAL LIB.		325	Financial and Statistical analysis
	INFO CENTRE		2,000 –20,000	Full-screen entry, display & multi-dimensional analysis. Interfaces to other I.C. products. Runs under VM VSAPL on IBM mainframes.
	APL Spreadsheet Manager		195	APL spreadsheet – links to popular spreadsheet software.
	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.
	MICROSPAN	250	Comprehensive APL tutor
	LOGOL	poa	Logistics management system for PC, Forecasting, Inventory Control, Scheduling, Distribution, etc
MetaTechnics	MetaScreen	99	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.
	APL-IEEE488	99	Controls IEEE488/GPIB Bus from APL*PLUS/PC.
	PLOT/PC	99	2D & 3D Graphics package. Includes interactive diagram Editors.
	Browse	99	Scrolling of DOS files, large APL variables.
	ADAPTA DLS	poa	Production & purchasing scheduling for process manufacturing.
	ADAPTA MSP	poa	Job-shop loading & scheduling for multi-stage production.
MicroAPL	MicroTASK	250	Product development aids
	MicroFILE	250	File utilities and database
	MicroPLOT	250	Graphics for HP plotters etc
	MicroLINK	250	General device communications
	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
	PC TOOLS Vol 1	295	
	STATGRAPHICS Rel 1	495	
	STATGRAPHICS Rel 2	535	
Parallax	ExecuCalc	$5,000	Mainframe-based electronic spreadsheet for VM/CMS & MVS/TSO with links to micro products.
	ExecuPlot	$5,000	Mainframe-based colour graphics 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
Sugar Mill	Stat 1	$129.95	Statistical toolbox, menu driven
Uniware	*Mainframe*		
	STSC's ENHANCEMENTS	10,715	Quad-functions & nested arrays for IBM VSAPL under VM/CMS and MVS/TSO
	STSC's SHAREFILE	10,715	Component files for IBM VSAPL under VM/CMS and MVS/TSO and for IBM APL2
	PROGRAMMER TOOLS & UTILITIES	5,715	
	FILEPRINT	5,715	
	FILESORT	5,715	
	FILECONVERT	5,715	
	FILEMANAGER (EMMA)	5,715	STSC's database package.
	APL*PLUS COMPILER	21,430	First APL compiler. Complements APL*PLUS enhancements and Sharefile under VM/CMS and MVS/TSO.
	EXECUCALC	3,995	Mainframe spreadsheet compatible with VISICALC and part of LOTUS 1-2-3 under VSAPL (VM or TSO).
	Microcomputer		
	STATGRAPHICS	725	Statistics & Graphics for PCs.
	STATGRAPHICS FCA	140	An add-on module to STATGRAPHICS: Factorial Correspondence Analysis.
	APL*PLUS/PC TOOLS		
	VOL1	325	Incl. 327 x IRMA support, RAM disk, full screen data entry, menu input, report generation, games.
	VOL2	125	Incl. File documentor, screen editor, exception handling.
	SPREADSHEET MNGR	250	APL spreadsheet with built-in ASCII, LOTUS and SYMPHONY interfaces.
	APL*PLUS/PC FIN. & STAT. LIBRARY	350	Collection of financial and statistical utilities.
	POCKET APL	140	Smaller version of APL*PLUS/PC.
	UNIASM	275	Collection of assembler routines for APL*PLUS/PC users.
	UNITAB^{tm}	240	APL*PLUS/PC spreadsheet-like data entry and validation system.
	The APL DEBUGGER^{tm}	105	First released APL*PLUS/PC debugger.
	OVERLAYS	250	Fast assembler routines to handle overlays in APL*PLUS/PC.
	R:BRIDGE	380	Interface between APL*PLUS/PC & R:BASE 5000.
	DMA	380	A version of EMMA (APL database manager) for APL*PLUS/PC users.
	APL2C	295	Interface between APL*PLUS/PC and DATALIGHT C language.
	ADAPTA/DLS	33,333	Production & purchasing scheduling for process manufacturing.

APL CONSULTANCY

(prices quoted are per day unless otherwise marked)

COMPANY	PRODUCT	PRICES £	DETAILS
APL Consultancy	Consultancy	poa	Project management, financial applications, relational databases. Difficult problems solved. Management consultancy. Links to non-APL systems. From consultant level to managing consultant. Documentation a speciality.
APL Software Technology	Consultancy	poa	Technical & business systems, micros, networking & communications a speciality.
Boeing	Consultancy	poa	
Camacho	Consultancy	poa	Specialising in programming & manual writing.
Chapman	Consultancy	150-300	24-hour programmer: APL, C, assembler, graphics; PC, mini, mainframe, network.
Cocking/Drury	Consultancy	120-150 140-200 185-300 275-400	Junior consultant Consultant Senior consultant Managing consultant
Delphi	Consultancy	poa	Specialising in management reporting systems and APL on microcomputers.
Dyadic	Consultancy	poa	APL system design, consultancy, programming & training for Dyalog APL, VSAPL, APL*PLUS, IPSA APL etc.
E & S	Consultancy	150 –250	System prototyping: all types of information system.
FASTCODE	Consultancy	poa	Specialise in improving performance of APL applications on micros & mainframes.
Gen. Software	Consultancy	100+	
H.M.W.	Consultancy	100-250	System design consultancy, programming.
Inner Product	Consultancy	200	On-site micro-mainframe APL, PC/DOS & Assembler
Lloyd Savage	Consultancy	poa	Decision support, particularly specialising in Sales & Marketing systems.
M.B.T.	Consultancy	poa	
Mercia	Consultancy	poa	APL*PLUS & VSAPL consultancy.
MetaTechnics	Consultancy	poa	Management Information & Production. Engineering APL – C/Assembler custom programming
MicroAPL	Consultancy	poa	Technical & applications consultancy.
M.T.I.	Consultancy	poa	Specialise in Maintenance and development of existing APL systems
Parallax	Consultancy	$750	Introductory APL, APL for End-user & Advanced Topics in APL
QB On-Line	Consultancy	200	Specialising in Banking, Financial & Planning Systems.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL
I.P. Sharp	Consultancy	poa	Consultancy & support service world-wide.
Peter Cyriax Systems	Consultancy	100-150 120-200 160-300	Junior Consultant Consultant Senior Consultant
Uniware	Consultancy	call	Junior to managing consultancy in APL.

APL TRAINING COURSES
(Prices quoted are per course unless otherwise stated)

COMPANY	PRODUCT	PRICES £
Cocking/Drury	3 day APL Fundamentals	375
	4 day APL*PLUS/PC Intermediate	525
	5 day APL System Design	595
	4 day Introduction to APL2 (in-house)	2,500+
	4 day APL2 in Depth (in-house)	2,500+
Inner Product		poa
M.B.T.		poa
Mercia	3 day Introduction to APL*PLUS/PC	350
	(in-house)	1,250
	2 day APL*PLUS/PC Enhancements	240
	(in-house)	800
	3 day APL*PLUS System Design	375
	(in-house)	1,200
Parallax		poa
Uniware	Courses	
	5 day Introduction to APL	call
	5 day Advanced APL	call
	2 day Get the best from STATGRAPHICS	call
	5 day Get the best from R:BASE 5000	call

OTHER PRODUCTS

COMPANY	PRODUCT	PRICES £	DETAILS
APL People	Employment Agency	poa	Permanent employees placed at all levels. Contractors supplied for short/long-term projects, supervised.
Mercia	ALLCARD	495+	Memory management unit, allowing 952K under DOS – extra 312K APL*PLUS/PC workspace.
	MULTI-APL	195+	Multi-task/Multi-user/Network APL*PLUS/PC with file locking, etc.
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.

VENDOR ADDRESSES

COMPANY	CONTACT	ADDRESS & TELEPHONE No.
Analogic Corporation	Denise Favorat	8 Centennial Drive, Centennial Industrial Park, Peabody, Mass. U.S.A. 01961 ☎ 617-246-0300
APL 385	Adrian Smith	Brook House, Gilling East, York. ☎ 04393-385
APL Consulting	Jill Moss	17 Barton Street, Bath, Avon BA1 1HQ ☎ 0225 62602
APL People	Valerie Lusmore	17 Barton Street, Bath, Avon. ☎ 0225-62602
APL Software Ltd	Philip Goacher	27 Downs Way, Epsom, Surrey KT18 5LU ☎ 03727-21282 17 Barton Street, Bath, Avon BA1 1HQ ☎ 0225-62602
APL Software Technology (UK) Limited	John Hagger Per Hultin	14 Rosewood Avenue, Alveston, Bristol BS12 2PP ☎ 0454 415737 46 Vicarage Road, South Benfleet, Essex SS7 1PB ☎ 03745 50501

Beta-plan APS	Kim Andreasen	Stengrade 75, DK-3000 Helsingor, Denmark. ☎ 45 2 21 48 48
Boeing Computer	Suzanne Hunt	P.O. Box 747, 364 Euston Road, London NW1 3BQ
Butel Technology Ltd.	Mike Munro	Butel House, 3 Great West Rd., London W4 5QJ ☎ 01-995-1433
Anthony Camacho		2 Blenheim Road, St. Albans, Herts AL1 4NR. ☎ St. Albans 60130
Paul Chapman		18 Trevelyan Road, London, SW17 9LN ☎ 01-767 4254
Cocking & Drury Ltd.	Romilly Cocking	16 Berkeley Street, London W1X 5AE. ☎ 01-493 6172 155 Friar Street Reading RG1 1HE. ☎ 0734-588835
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	Park House, The High Street, Alton, Hampshire. ☎ 0420-87024
E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS. ☎ 0689-24741
Farnell International Instruments Ltd.	R. Fairbairn or Roger Attard	Jubilee House, Sandbeck Way, Wetherby, W. Yorks. ☎ 0937-61961 Davenport House, Bowers Way, Harpenden, Herts. ☎ 05827-69071
FASTCODE	Andrew Dickey	P.O. Box 281, Croton-on-Hudson, New York 10520, U.S.A. ☎ (914) 271-3200
General Software Ltd.	M.E. Martin	22 Russell Road, Northolt, Middx. UB5 4QS. ☎ 01-864 9537
H.M.W. Programming Consultants Ltd.	Ken Jackson	142 Feltham Hill Rd, Ashford, Middx. 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	Chris Sell	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
Lloyd Savage Ltd	Philip Johnson	Cambridge House, Oxford Road, Uxbridge. Middx, UB8 2UD. ☎ 0895-59826
Mercia Software Ltd.	Gareth Brentnall Barrie Webster	Aston Science Park, Love Lane, Birmingham B7 4BJ. ☎ 021-359 5096
MetaTechnics Systems Ltd	John Stenbridge David Toop	Unit 216, 62 Tritton Road, London, SE21 8DE. ☎ 01-670 7959
MicroAPL Ltd.	Bernadette Leverton	19 Catherine Place, London SW1E 6DX ☎ 01-834 9022
Mine of Information	Richard Ross-Langley	PO Box 1000, St. Albans, Herts AL3 6NE. ☎ 0727 52801
Modern Business Technology Ltd. (MBT)	Michael Branson	P.O. Box 87, Guildford, Surrey GU4 8BB ☎ 04868-23956
M.T.I.	Ray Cannon	7 Pine Wood, Sunbury-on-Thames, Middx. TW16 6SH ☎ 09327 80848
Parallax Systems Inc.	Kevin Weaver	60 West 9th Street, New York, New York 10011, U.S.A. ☎ 212-475-4001
Peter Cyriax Systems	Peter Cyriax	213 Goldhurst Terrace, London NW6 3ER ☎ 01-624 7013 (Answerphone) 0860-337963 (Mobile)
Portable Software	Richard Smith	60 Aberdeen Ave, Cambridge, Mass. U.S.A. 02138. ☎ 617-547-2918
QB On-Line Systems	Philip Bulmer	5 Surrey House, Portsmouth Rd Camberley, Surrey, GU15 1LB. ☎ 0276-20789
The Rochester Group	Robert Pullman	164 Pinnacle Rd., Rochester NY 14620 ☎ 716-461-3169
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
Sugar Mill Software Corp.	Lawrence H. Nitz	1180 Kika Place, Kailua, Hawaii 96734 ☎ (808) 261-7536
Tektronix UK Ltd.	Paul Morgan	Fourth Avenue, Globe Park, Marlow, Bucks SL7 1YD. ☎ 06284-6000
Uniware	Eric Lescasse	15 Rue Erlanger – 75016 Paris – France ☎ (1) 45-27-20-61 Telex: 648348F UNIWARE

Info Center/1

an IBM licensed program that helps business professionals perform their daily tasks quickly and productively

Info Center/1 provides an integrated, multifunction information center environment compatible with predecessor products such as ADRS II and APLDI II. A full-screen interface, with prompts and extensive help facility, provides easy access to the following powerful general business functions, as well as providing the full power of APL:

Query System

The Query System provides a simple, effective way to interactively access, analyze, manipulate, and report information stored in files of up to several hundred megabytes.

Reporting System

Provides an organization with a single, comprehensive system for generating and maintaining reports. Standard calculations can be defined and stored for future use. Calculations can be made with predefined functions and with APL.

Data Entry and Validation

This tool allows information center personnel to tailor panels for users to display, update, and enter data in column format.

Financial Planning System

The Financial Planning System provides a set of 60 modeling routines that work with the Reporting System and address periodic data. Some examples are:

- Financial analyses and plans
- Statistical analyses and projections
- What if analyses and modeling
- Project evaluations and risk analyses.

Business Graphics

The Business Graphics facility is a particularly powerful yet flexible tool for interactively producing the following types of charts: line graphs, surface charts, histograms, pie charts, scatter plots, bar charts, stacked bar charts.

Technical Data

Info Center/1 is an IBM Licensed Program, Program Number 5668-897.
The program runs under CMS and TSO together with the following IBM programs or their equivalents: APL2 or VS APL, Application Prototype Environment, GDDM (Graphical Data Display Manager).
Some examples of terminals supported are: IBM 3277, 3279, 3270 PC/G and GX.

IBM

RECENT MEETINGS

This section of VECTOR is intended to document the seminars delivered at recent meetings of the Association, particularly for the benefit of those members based away from London who often find it hard to find the time to attend. It also covers other selected events which are likely to be of interest to the wider APL community.

We are dependent on the willingness of speakers to provide us with a written version of their talk, and we would remind them that "a picture's worth a thousand words". Copies of slides and transparencies will enhance their articles.

The Activities Officer (see inside back cover) will respond enthusiastically to offers from individuals who wish to contribute seminars and supporting papers.

Recent meetings

Our coverage of recent meetings in this issue is limited to the paper given by Dick Bowman, of C.E.G.B., last November, looking at APL's links with the outside world.

On Friday 20th February, the topic of the B.A.A. meeting at the Royal Overseas League was 'Are Your Systems Reliable'. The two speakers, Linda Kindred from the Wellcome Foundation and Chris Campen from the British Airports Authority, gave their views on good systems development practice through quality assurance. Their talks were followed by a panel discussion. A full report of this meeting will appear in the next issue of VECTOR.

A number of people expressed an interest in the Quality Assurance Forum mentioned by Linda Kindred during her talk. Anyone interested in finding out more about it should contact:

Gordon Irving,
17 St Catherine's Road,
Ruislip,
Middlesex, HA4 7RX.
Tel: 0895 635222

The Outside World

by Dick Bowman

A feature becoming commonplace in newer APLs is the ability to use routines written in other languages. The object of this paper is to review some limited experience in using these features, suggest where the benefits to APL lie and pose some questions about the applicability of this sort of link.

There is no pretence that this is an exhaustive survey – it is merely the collection of some recent experience.

1. APL*PLUS/PC Assembler Functions

An expected corollary of the ⎕CALL system function is that use of assembler routines in appropriate places should lead to some speedup in application processing. Certainly use of the <CSSMATML> routine described in the STSC documentation and incorporated in their distributed <INPUT> workspace had helped lead to this expectation.

Some experimentation with the ASMFNS distributed workspace was inconclusive. There was a measurable performance increase with test data but in the context of an application (where function calls were frequent, but arguments were brief) the achieved improvement was judged unjustifiable in contrast to the effort which would have been required to implement the necessary changes. My problem seemed to centre around wanting to use routines subtly different from those provided in pre-written form; not having the patience or skills to write my own Assembler routines I burdened those that I used down with APL prologues and epilogues converting data between what I had and what was wanted.

Questions:

- Does anyone have a good reliable way of measuring performance on the IBM PC – can we see quantification of the speed gains achievable?
- How much effort is required to learn enough Assembler to be able to cost-effectively integrate APL and Assembler if performance is the only criterion?

2. IBM ICU

In their manuals IBM provide an example of a function which can invoke ICU to produce routine 'business' graphics diplays; it's a little rudimentary, suffering lack of flexibility. Straightforward generalisation produces the <CHART> function (see Figure 1); a little more work results in <CHARTFREE> (Figure 2) which handles 'free data' (to use the vernacular). A description and examples of use are shown in Figure 3; the most useful mode of operation, of course, is with pre-defined chart formats.

For all its faults – the main one being that there seems to be absolutely no scope to extend its capabilities – being able to link direct to a pre-written graphics package (and one used by lots of people in non-APL applications) carries obvious benefits.

One notes with interest that future plans for ICU appear to be along the lines of integrating more capability into it; in the context of the infamous 'line smoothness – enter a number from 1 to 99', one wonders whether a more desirable evolution path would be as an appendage to programming systems with flexibility and rigour than for ICU to grow into yet another 'all-purpose' system with its own rules, idiosyncrasies and expertise.

3. APL2 ⎕NA

APL2 Release 2 introduced the facility to call programs written in other languages, Assembler and Fortran (with some restrictions) from both CMS and TSO, REXX from CMS only (my personal experience is limited to use within TSO). The potential is very attractive – by contrast to previous excursions by IBM APLs into the outside world, the implementation was surprisingly clean. The most awkward part of the process was to come to terms with the format of the NAMES file which provides the cross-relation between APL variables and Fortran/Assembler arguments and results – the best advice I can offer is to work your way into it gradually with programs which you are familiar with and which are under your control.

Once you're over the hurdle of setting up properly-formed NAMES file entries (more plentiful examples in the documentation would help) usage is simplicity personified.

For example, given a Fortran subroutine which we want to call <NELLY> all that's necessary is to type:

```
3 11 ⎕NA 'NELLY'
```

You now effectively have a locked function named <NELLY> which (assuming that the Fortran routine has been properly link-edited into a library which you have allocated) you can treat just as if it had all along been written in APL. The left argument to ⎕NA is presently a little over-elaborate (the 3 means that you want to create a function; the 11 that your victim is a Fortran or Assembler routine); one awaits future extension with great anticipation. Having the function not only can you use it, you can move it around with)LOAD,)COPY and)PCOPY.

Having got the feature, and knowing how to use it, what does it give in return? Essentially three things:

a) Doing Things Faster

   This is the obvious payoff – given a utility function you can avoid interpreter overhead. Equally obviously it should be possible to measure the performance gain; a place to start is the example provided in the IBM manuals. Compiling and linking were straightforward (we're using MVS/XA and I found that it was only necessary to specify AMODE(31) at the link-edit stage, which is a slight departure from the official word, and may well be installation-sensitive. As an aside, the state of the Fortran art vis-a-vis Fortran66/Fortran77/VSFortran/XA is a veritable quagmire – I've (temporarily?) adopted the ultra-conservative line of compiling and link-editing what I need myself rather than rely on standard libraries, at least I have only myself to blame.

   Comparison with equivalent APL was that with vector arguments up to 10 elements there was a very slight advantage to the Fortran which steadily widened as the argument length increased until at 1000 elements the Fortran was about 2.5 times faster. Bear in mind that:

   i) You have to do a little preparatory work in APL setting up arguments that the Fortra can swallow (this is universal and can lead to APL being a better performer tha compiled code in the right, and not uncommon, circumstances).

ii) The Fortran routine (as written) has an upper bound on argument length which is not there for the APL (within reason).

One could be contentious at this point and ponder the relative values of APL compilers against making wider use of facilities like this – but I don't really think that 'doing it faster' is in any way the major attraction of the feature.

b) Doing Things Better

This is what gets much closer to the point; there's a vast legacy of pre-written code in other languages which performs well-defined tasks to a predictable standard. We don't have to rewrite it all in APL and we can be confident of accuracy. Given the CEGB situation as a case in point – here's a list of bought-in software which is available to anyone developing Fortran software on the system:

NAG Library	– Mathematical/Scientific Subroutines
NAG Graphics	– Graphical Supplement
Harwell Library	– AERE main software library
FACSIMILE	– Ordinary Differential Equations
EISPACK	– Eigenvalue Analysis Solutions
REDUCE	– Computer Algebra
DASL	– Data Approximation Subroutines
MINITAB	– Interactive Statistics
IBMSSP	– IBM Scientific Subroutines
GLIM	– Statistical model building
GENSTAT	– Statistical model building

It's nothing like a complete list, libraries like NAG contain hundreds of routines themselves, and it's all available now to the APL user as well. Why does this matter? Here's an example:

The Generating Board owns many cooling towers at power stations; they're hundreds of feet high and have walls of reinforced concrete less than six inches thick. The problem is to deduce a complete wall profile from accurate measurement at a few select positions; we had a Fortran program which could do this and was of satisfactory accuracy, but it didn't return the figures in a machine-readable format which we could use for further analysis; we needed a new program of comparable accuracy and for a variety of reasons APL was the preferred development route.

The obvious technique to use was a spline-fit (which was also what our existing program used); immediate candidates were in IBM's GRAPHPAK workspace and Gilman and Rose. But the problem was that neither of these got within an acceptable margin of our original program (which had had exhaustive checks for accuracy). Up to this point one had had a certain amount of faith in the accuracy of these routines.

NAG offered a pair of routines (E01BAF and E02BBF) which seemed to fit the bill (they are intended for use as a pair); Figure 4 shows them in context of being used (most of the activity in the APL is to marshall the various arguments and set up space for results). It also points up the shortcoming of such a lot of scalar programming; E02BBF is a one-shot subroutine and you loop around (I guess "each" might be useable by the brave – but I was under time constraints to deliver).

At the end of the day results agree well with our target performance.

Another example using NAG routines is the <POLYFIT> function shown in Figure 5 – what makes this relevant being that it provides the solution to the APL2 "domino" problem which I raised in an earlier issue. Whereas a naively-coded APL2 function failed to fit a curve after order 2, with VSAPL and APL*PLUS/PC carrying on to order 5 – on the specific data in the previous discussion <POLYFIT> was willing to produce results all the way up to order 10. To save effort with pencil and graph paper, Figure 6 shows a calculated third-order fit to the data.

Again some oddities that you find when you use pre-written code: <E02ADF> is happy to provide an array result giving coefficients all the way up to your desired order, whereas the closely related <E02AEF> resolutely insists on taking single x-values only.

Note that the passing of time since writing <RCALC> has led to a slightly more confident coding style.

Question – How much APL code is 'good enough' and how much fails if accuracy constraints are rigorous?

c) Doing Things You Couldn't Otherwise

Again an example:

We make extensive use of Tektronix (and other) graphics terminals, which operate in response to sequences of ASCII characters. Back in the old days of VSPC/VSAPL this was no problem; type:

```
)VSPC PUNCH CONTROL
```

and what you send arrives at the terminal untranslated, unadulterated and uncontaminated.

Migration to TSO/APL2 brought BIG TROUBLE (parenthetically, this is the only conversion problem which has ever worried us) – if it wasn't printable it got mangled. We had an investment in graphics hardware and at the time (this is now changing dramatically) some reluctance to get involved with IBM 'graphics' terminals (because they really weren't graphics terminals).

Some IBM conversion manuals told us where the problem came from ('a TPUT with EDIT') and led us off in the general direction of writing an AP which would bypass the problem. We never really got anywhere (our friendly SE offered to sell us a few 3277/GAs) and APL2 Release 2 came along, accompanied by an unattributed verbal remark that ⎕NA made our problem trivial. We didn't really see this, but there was also an offer of IBM sorting the problem out for us at a fee of £77 a manhour; we figured that if knowing this sort of thing let you earn £77 an hour we'd like some of the action for ourselves.

Fortunately (?) we have a well-endowed Fortran system, and the Fortran graphics people had got round the problem themselves some years earlier; so what we did was to use their solution and ⎕NA to reach it. The outcome is shown in Figure 7; <DRAWABS> is a device-independent function which draws a line to a specified point.

With device type specified in global <dev>, function <DRAWABST4107> constructs specific code for the Tektronix 4107 terminal; it gets translated to ASCII by the <ASCIIOUT> function before finally being chopped into 256-byte (maximum imposed by the assembler routine) chunks by <OUT> for transmission to the terminal by external function <VTADEO31>. Apologies for the convoluted route, the bug(s) and the extraneous code. Note again the imposition of finite size limits and the consequent loop potential of the APL code (in practice this application very rarely wants to send a string of more than 256 characters in one go – ideally we should also do some blocking for transmission, but we don't).

The associated Assembler code is shown in Figures 8 and 9; there is some allegation that one or two of the translate table entries are incorrect.

At the end of the day it really was trivial, but we didn't ever get to see the £77 either.

4. APL2)EDITOR

IBM Reference documents for APL2 contain sketchy comment about use of 'any operating system editor that you choose to use' for function and operator editing; curiosity led to investigation of this feature. The IBM literature seems somewhat imprecise – what follows here is an attempt to provide a specific example of use of the feature in one specific installation; the environment is TSO with Release 1 of ISPF, the objective is to use the ISPF editor to edit an APL function.

a) Augment standard CLISTs with a few of your own; Figure 10 shows the necessary TSO allocation and this is best done before invoking APL. There are two new CLISTs involved, called <ENT> and <CEDIT>, shown in Figures 11 and 12 – the IBM manuals lead one to the impression that <CEDIT> would suffice, but this wasn't the case; <ENT> is excessively complex and repeats some dataset allocation needlessly – what you see here is something which works rather than a honed-down version.

b) Specify the name of the editor you want to use with the)EDITOR command – in this instance the name is <ENT>

c) Edit as required; a typical display is shown in Figure 13.

Having gone to all this trouble, the results are none too exciting; first intimation of trouble being the time elapsing between wanting to edit and being able to.

Very ad-hoc timing comparisons for a simple editing exercise gave the following results:

```
        )EDITOR 1               29 ms
        )EDITOR 2              235 ms
        )EDITOR ENT            989 ms
```

An obvious candidate for speedup is the excessive amount of work in the <ENT> CLIST; some experimentation with chopping this about (and also taking the hint in the IBM manual) was none too conclusive; my personal resolution has been to avoid the facility until either someone shows me that it can be made to work satisfactorily or I can find a strong reason for needing to use the ISPF editor to edit APL objects.

Further reflection leads me to the opinion that my <MYALLOC> and <ENT> CLISTs are pretty awful, but I remain dubious that refinement would lead to this route becoming my preferred editor – I feel that the overhead of creating and using data files must slow things up ultimately. But I stress that this is all based on very limited experience in one environment – Phil Last tells me that use of the CMS editor is preferable to)EDITOR 2 in a CMS environment (again with a proviso or two); I suspect that if I were invoking APL2 from within ISPF (rather than the other way about) there could be a different conclusion.

Question – Aside from the option of newcomers to APL using a familiar editor what is this feature for?

Acknowledgements:
My colleagues John Dedman and David Leal for their patience in the face of my ignorance of Fortran programming and ISPF; Gill Darbyshire for doing the spadework on <POLYFIT> as a first task in APL.

References:

1. *STSC APL*PLUS/PC System Reference Manual*
2. *APL2 Programming: Language Reference*; IBM Corporation; SH20-9227
3. *APL2 Programming: System Services Reference*; IBM Corporation; SH20-9218
4. *APL2 Programming: Using the Supplied Workspaces*; IBM Corporation; SH20-9233
5. *APL – An Interactive Approach*; Gilman and Rose; Wiley
6. *NAG Fortran Library Manual*; Numerical Algorithms Group
7. *VECTOR Vol 3 No 1*, Page 99; Technical correspondence.

Figure 1. The <CHART> function

```
    ∇ CHART W;CHRTCTL;CTL;DAT;DATACTL;HEADING;KEYS;LABELS;RC;X;Y;⎕IO
      ⎕IO←1
      →(2∧.=RC←126 ⎕SVO 2 3 ⍴'CTLDAT')/SHAREOK
      'AP126 NOT SHARING'
      →0
     SHAREOK:X←2⊃W
      Y←(¯2↑ 1 1 ,⍴Y)⍴Y←3⊃W
      CHRTCTL←76⍴' '
      CHRTCTL[⍳4]←4 IO 0                    ⍝ VERSION OF ICU
      CHRTCTL[4+⍳4]←4 IO 2                  ⍝ ICU FUNCTION REQUIRED
      CHRTCTL[8+⍳4]←4 IO 0                  ⍝ PFKEY INFORMATION DISPLAY
      CHRTCTL[12+⍳4]←4 IO 0                 ⍝ SAVE/RESTORE PERMITTED
      CHRTCTL[16+⍳8]←8↑1⊃W                  ⍝ NAME OF SAVED FORMAT
      CHRTCTL[25]←'*'                       ⍝ NAME OF SAVED CHART DATA
      CHRTCTL[32+⍳4]←4 IO 0                 ⍝ TIED OR FREE DATA
      CHRTCTL[36+⍳4]←4 IO 1↑⍴Y              ⍝ NUMBER OF COMPONENTS
      CHRTCTL[40+⍳4]←4 IO⍴X                 ⍝ NUMBER OF X ELEMENTS
      CHRTCTL[44+⍳4]←4 IO ¯1↑⍴5⊃W           ⍝ LENGTH OF EACH KEY
      CHRTCTL[48+⍳4]←4 IO ¯1↑⍴6⊃W           ⍝ LENGTH OF EACH LABEL
      CHRTCTL[52+⍳4]←4 IO⍴HEADING←4⊃W       ⍝ LENGTH OF CHART HEADING
```

```
[20]  CHRTCTL[57]←'*'                          ⍝ PRINTER OR GDF FILE NAME
[21]  CHRTCTL[64+⍳12]←,4 IO 0 80 2             ⍝ PRINTER DEPTH, WIDTH, COPIES
[22]  DATACTL←0⍴0
[23]  KEYS←,5⊃W
[24]  LABELS←,6⊃W
[25]  DAT←CHRTCTL,KEYS,LABELS,HEADING
[26]  Y←,Y
[27]  CTL←¯10,(⍴CHRTCTL),(⍴DATACTL),DATACTL,(⍴X),X,(⍴Y),Y,(⍴KEYS),(⍴LABELS),⍴HEADING
[28]  RC←CTL
    ∇
```

Figure 2. The <CHARTFREE> function

```
    ∇ CHARTFREE W;CHRTCTL;CTL;DAT;DATACTL;HEADING;KEYS;LABELS;RC;X;Y;⎕IO
[1]   ⎕IO←1
[2]   →(2∧.=RC←126 ⎕SVO 2 3 ⍴'CTLDAT')/SHAREOK
[3]   'AP126 NOT SHARING'
[4]   →0
[5]  SHAREOK:X←2⊃W
[6]   Y←(¯2↑ 1 1 ,⍴Y)⍴Y←3⊃W
[7]   CHRTCTL←76⍴' '
[8]   CHRTCTL[⍳4]←4 IO 0                       ⍝ VERSION OF ICU
[9]   CHRTCTL[4+⍳4]←4 IO 2                     ⍝ ICU FUNCTION REQUIRED
[10]  CHRTCTL[8+⍳4]←4 IO 0                     ⍝ PFKEY INFORMATION DISPLAY
[11]  CHRTCTL[12+⍳4]←4 IO 0                    ⍝ SAVE/RESTORE PERMITTED
[12]  CHRTCTL[16+⍳8]←8↑1⊃W                     ⍝ NAME OF SAVED FORMAT
[13]  CHRTCTL[25]←'*'                          ⍝ NAME OF SAVED CHART DATA
[14]  CHRTCTL[32+⍳4]←4 IO 1                    ⍝ TIED OR FREE DATA
[15]  CHRTCTL[36+⍳4]←4 IO⍴X                    ⍝ NUMBER OF COMPONENTS
[16]  CHRTCTL[40+⍳4]←4 IO⍴∊X                   ⍝ NUMBER OF X ELEMENTS
[17]  CHRTCTL[44+⍳4]←4 IO ¯1↑⍴5⊃W              ⍝ LENGTH OF EACH KEY
[18]  CHRTCTL[48+⍳4]←4 IO ¯1↑⍴6⊃W              ⍝ LENGTH OF EACH LABEL
[19]  CHRTCTL[52+⍳4]←4 IO⍴HEADING←4⊃W          ⍝ LENGTH OF CHART HEADING
[20]  CHRTCTL[57]←'*'                          ⍝ PRINTER OR GDF FILE NAME
[21]  CHRTCTL[64+⍳12]←,4 IO 0 80 2             ⍝ PRINTER DEPTH, WIDTH, COPIES
[22]  DATACTL←∊⍴ ¨ X
[23]  KEYS←,5⊃W
[24]  LABELS←,6⊃W
[25]  DAT←CHRTCTL,KEYS,LABELS,HEADING
[26]  X←∊X
[27]  Y←∊Y
[28]  CTL←¯10,(⍴CHRTCTL),(⍴DATACTL),DATACTL,(⍴X),X,(⍴Y),Y,(⍴KEYS),(⍴LABELS),⍴HEADING
[29]  RC←CTL
    ∇
```

Figure 3. How to use <CHART>

The <CHART> function provides a link to the IBM CHART utility.

Prerequisite: If you wish to save or retrieve chart format definitions you must have a file called <tsoid.CHART.ADMCFORM> allocated to the ddname <ADMCFORM>. This is done automatically if the workspace is used stand-alone (see <LX > for specifics).

Note that terminology is CHART-terminology; for a full description of facilities consult the relevant IBM manuals.

To draw a chart:

CHART name xvalues yvalues heading key labels

where:

<name> – either '*' (do not use a stored chart format) or name of stored chart format

<xvalues> – vector of xvalues (at present this link is restricted to 'tied' data)

<yvalues> – either vector of yvalues
or matrix with each row of the matrix a set of yvalues
Missing values are enterable as 1E72.

<heading> – Overall title for the chart. May be elided by entering as ''

<key> – Chart key (matrix with one row for each key). May be elided by entering as ''.

<labels> – Chart labels (matrix with one row for each label). May be elided by entering as ''.

For example:

```
NAME←'*'
X←⍳5
Y←5⍴?100
HEADING←'FIVE RANDOM NUMBERS'
KEY←1 6⍴'RANDOM'
LABELS←5 1⍴'ABCDE'
CHART NAME X Y HEADING KEY LABELS
```

Function <CHARTFREE> provides an experimental equivalent for free data. Syntax of usage is the same, but both <xvalues> and <yvalues> are nested vectors; for example:

```
X←(1 2 3) (4 5 6 7)
Y←(?3⍴100) (?4⍴200)
CHARTFREE '*' X Y '' '' ''
```

Figure 4. <RCALC> – using E01BAF and E02BBF

```
    ∇ Z←RCALC;M;X;Y;K;C;LCK;WRK;LWRK;IFAIL;I;RES
  ⍝ R J BOWMAN - CALCULATE SPLINE-FIT RADIUS AT REQUIRED HEIGHTS
   M←(⍴ZACT)
   X←ZACT
   Y←RACT
   K←(4+M)⍴0.1
   C←(4+M)⍴0.1
   LCK←4+M
   WRK←(16+6×M)⍴0.1
   LWRK←16+6×M
   IFAIL←0
   RES←E01BAF M X Y K C LCK WRK LWRK IFAIL
   K←⊃RES[1]
   C←⊃RES[2]
   I←0
   Z←⍳0
 NEXT:→((I←I+1)>⍴ZREQ)↑0
   →(ZREQ[I]>⌈/ZACT)↑ERR
   S←0.1
   Z←Z,⊃(E02BBF(4+M)K C(ZREQ[1])S IFAIL)[1]
   →NEXT
   ERR(Z←Z,0
   →NEXT
    ∇
```

Figure 5. The <POLYFIT> function

```
    ∇ Z←DEG POLYFIT XY;A;I;IFAIL;K;NEXTL;P;S;W;W1;W2;X;Y;⎕IO
   ⍝POLYNOMIAL FIT
    ⎕IO←1
    K←1+DEG
    (X Y)←XY
    W←(⍴X)⍴1                       ⍝ WEIGHTS
```

```
      W1←(3,ρX)ρ0                    ⍝ REQUIRED WORK AREA
      W2←(2,K)ρ0                     ⍝ REQUIRED WORK AREA
      A←(2ρK)ρ0                      ⍝ RESULTS AREA [1]
      S←Kρ0                          ⍝ RESULTS AREA [2]
      IFAIL←0                        ⍝ RESULTS AREA [3]
      (A S IFAIL)←E02ADF(ρX)K K X Y W W1 W2 A S IFAIL
      A[A[;K]                        ⍝ REQUIRED COLUMN ONLY
      I←1
      Z←(ρX)ρ0
      X←X XCAP⍳ρX                    ⍝ NAG 'NORMALISATION'
      NEXTL←((ρX)ρNEXT),0
     NEXT:(P IFAIL)←0
      (P IFAIL)←E02AEF K A X[I]P IFAIL
      Z[I]←P
      →NEXTL[I←I+1]

    ∇

    ∇ Z←X XCAP I
     ⍝ 'NORMALISATION' - AS IN NAG MANUAL
      Z←((X[I]-⌊/X)-(⌈/X)-X[I])÷(⌈/X)-⌊/X
    ∇
```

Figure 6. The results of using <POLYFIT>

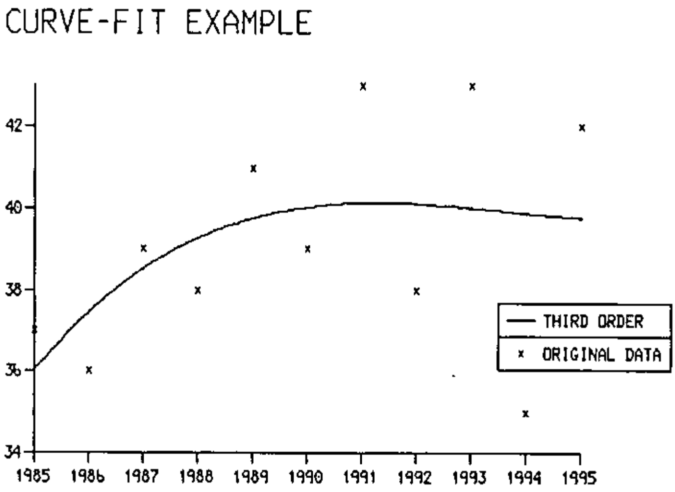


Figure 7. Achieving ASCII output to Tektronix 4107

```
    ∇ DRAWABS A;D
     ⍝ DRAW TO POSITION <A>
      OLDCP←CP
      CP←A
      A←CONNDC A
      ⍎'OUT ASCIIOUT (DRAWABS',DEV,' A),DTYPE↓NEWLINE'
      D←⎕DL IDLES×0.1
    ∇
    ∇ R←DRAWABST4107 A
[1]  ⍝ T4107-SPECOIFIC <DRAWABS>
[2]   R←ESC,'LG',T4107CODE A
    ∇

    ∇ Z←ASCIIOUT W
[1]  ⍝ CONVERSION FOR ASCII OUTPUT
[2]   Z←⎕AV[ASCIICON[⎕AV⍳W]]
    ∇

    ∇ OUT STRING;L
[1]  ⍝ SEND <STRING> TO TERMINAL
[2]
[3]   STRING←,STRING
[4]  NEXT:→(0=⍴STRING)↑0 ⍝ <VTADEO31> HAS INTERNAL 256-BYTE LIMIT
[5]   L←256⌊⍴STRING
[6]   VTADEO31 L(L↑STRING)
[7]   STRING←L↓STRING
[8]   →NEXT
    ∇
```

Figure 8. Assembler code to run VTADEO – part 1

```
************************************ TOP OF DATA **********************-CAPS ON-**
*--------CSECT VTADEO-------------------------------------------------
*********************************************************************
*        FILE VTADEO SET UP 11/4/86 FOR VTAM/NTO GRAPHICS           *
*        SIMILAR TO TCAM ADEOUT, EXCEPT DIFFERENT TRANSLATE         *
*        TABLES.                                                    *
*********************************************************************
VTADEO   CSECT
         DS    0H
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAV+4
         LA    13,SAV
         L     9,=V(VTRASA)
         L     5,=A(BUFFER)
         LM    7,8,0(1)               R7 IS A(LEN), R8 IS A(BUFFER)
         L     7,0(7)
         LR    0,7
LOOP     MVC 0(1,5),3(8)              PACK THE OUTPUT ARRAY
         LA    5,1(5)                 INCREMENT
         LA    8,4(8)
         BCT   7,LOOP                 TEST
         TR    BUFFER(256),0(9)       CONVERT FROM ASCII
         L     1,ADDR
         N     0,SIZE
         TPUT  (1),(0),R              REQUEST THE OUTPUT
         L     13,SAV+4
         LM    14,12,12(13)
         BR    14
SAV      DS    18F
ADDR     DC    X'02'
         DC    AL3(BUFFER)
BUFFER   DS    256C
SIZE     DC    X'0000FFFF'
         END
********************************* BOTTOM OF DATA *********************-CAPS ON-**
```

Figure 9. Assembler code to run VTADEO – part 2

```
************************************ TOP OF DATA **********************-CAPS ON-**
VTRASA   CSECT
         SPACE 15
*        THIS FILE IS VTRASA.                                   SRA 4/86
*         TRANSLATE TABLE FOR TSO/VTAM/NTO GRAPHICS OUTPUT
*
*                    0 1 2 3 4 5 6 7 8 9 A B C D E F
         DC     X'00010203372D2E2F1605250B0C0D0E0F' 0
         DC     X'101112133C3D322618193F271C1D1E1F' 1
         DC     X'404F7F7B5B6C507D4D5D5C4E6B604B61' 2
         DC     X'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3
         DC     X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4
         DC     X'D7D8D9E2E3E4E5E6E7E8E94AE05A5F6D' 5
         DC     X'79818283848586878889919293949596' 6
         DC     X'979899A2A3A4A5A6A7A8A9C06AD0A107' 7
         DC     X'00010203372D2E2F1605250B0C0D0E0F' 8
         DC     X'101112133C3D322618193F271C1D1E1F' 9
         DC     X'405A7F7B5B6C507D4D5D5C4E6B604B61' A
         DC     X'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F' B
         DC     X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' C
         DC     X'D7D8D9E2E3E4E5E6E7E8E979CF49B060' D
         DC     X'BF818283848586878889919293949596' E
         DC     X'979899A2A3A4A5A6A7A8A98B4F9B5F07' F
         END
********************************** BOTTOM OF DATA ********************-CAPS ON-**
```

Figure 10. TSO Allocation

```
************************************ TOP OF DATA **********************-CAPS ON-**
PROC 0                                                              00030000
 FREE F(SYSPROC)                                                    00360000
 ALLOC F(SYSPROC) DA('HCCK016.CLISTS',                    +         00450000
                     'HCCK.ISPF.LIBRN.CLIST',             +         00460000
                     'SYSC.TSOEA.CONTROL',                +         00470000
                     'SYS8.SPF.V1R1M0.CLIB',              +         00471000
                     'SYS8.ISR.V1R1M0.ISRCLIB')   SHR               00480000
********************************** BOTTOM OF DATA ********************-CAPS ON-**
```

Figure 11. CLIST <ENT>

```
*********************************** TOP OF DATA *********************-CAPS ON-**
PROC 1 DSN
 CONTROL MAIN MSG NOPROMPT
 EJECT
 CONTROL NOMSG NOPROMPT
 ERROR OFF
 SET &DSNAME = ISPF.FULLTSO.PROFILE.&SYSUID
 SET &DSVOL = TSM06
 SET &DSUNIT = 3350
 FREE F(ISPPROF)
 ALLOC FI(ISPPROF) DA('&DSNAME.') OLD
 FREE F(SPFK)
 ALLOC F(SPFK) DA(*)
 GO TO IPDF
 ALLOT: ERROR RETURN
 ALLOC F(SPFK) DA(*)
 FREE F(VK)
 FREE F(SYSPROC)
 FREE F(ISPLLIB)
 FREE F(ISPPLIB)
 FREE F(ISPMLIB)
 FREE F(ISPSLIB)
 FREE F(ISPTLIB)
 FREE F(ISPPROF)
 FREE F(SYSABEND)
 CONTROL MSG NOPROMPT
 ALLOC F(SYSPROC) DA('HCCK016.CLISTS',                 +
                     'HCCK.ISPF.LIBRN.CLIST',          +
                     'SYSC.TSOEA.CONTROL',             +
                     'SYS8.SPF.V1R1M0.CLIB',           +
```

Figure 11. CLIST <ENT> (continued)

```
                     'SYS8.ISR.V1R1M0.ISRCLIB')     SHR
 ALLOC F(ISPLLIB) DA('HCCK015.ISPF.LOAD',              +
                     'SYS8.ISR.V1R1M0.ISRLOAD',        +
                     'SYS8.ISP.V1R1M0.ISPLOAD')     SHR
 ALLOC F(ISPPLIB) DA('HCCK015.PANELS',                 +
                     'HCCK.ISPF.LIBRN.PANEL',          +
                     'SYS8.ISR.V1R1M0.ISRPLIB',        +
                     'SYS8.ISP.V1R1M0.ISPPLIB')     SHR
 ALLOC F(ISPMLIB) DA('HCCK015.MESSAGES',               +
                     'HCCK.ISPF.LIBRN.MESSAGE',        +
                     'SYS8.ISR.V1R1M0.ISRMLIB',        +
                     'SYS8.ISP.V1R1M0.ISPMLIB')     SHR
 ALLOC F(ISPSLIB) DA('HCCK015.SKELETON',               +
                     'HCCK.ISPF.LIBRN.SKELETON',       +
                     'SYS8.ISR.V1R1M0.ISRSLIB',        +
                     'SYS8.ISP.V1R1M0.ISPSLIB')     SHR
 ALLOC F(ISPTLIB) DA('SYS8.ISR.V1R1M0.ISRTLIB',        +
                     'SYS8.ISP.V1R1M0.ISPTLIB')     SHR
 ALLOC F(ISPPROF) DA('&DSNAME.') OLD
 ALLOC F(SYSABEND) DUMMY
 IPDF: ATTN OFF
 EJECT
 ISPSTART CMD(CEDIT)
 FREE F(ISPLLIB)
 FREE F(ISPPLIB)
 FREE F(ISPMLIB)
 FREE F(ISPSLIB)
 FREE F(ISPTLIB)
 FREE F(ISPPROF)
 FREE F(SYSABEND)
 CONTROL NOMSG NOPROMPT
 FREE F(SYSPOS)
 ALLOC F(SYSPOS) DA('TSOEASY.&SYSUID..DATA') SHR
 L2:ERROR RETURN
 CONTROL NOMSG NOPROMPT
 EXIT CODE (0)
 END
********************************** BOTTOM OF DATA ********************-CAPS ON-**
```

Figure 12. CLIST <CEDIT>

```
*********************************** TOP OF DATA *********************-CAPS ON-**
PROC 0
  ISPEXEC EDIT DATASET('HCCK016.APL2.EDIT')
********************************** BOTTOM OF DATA ********************-CAPS ON-**
```

Figure 13. The final result – editing an APL object

```
****** ****************************** TOP OF DATA ******************************
  Z←DEG POLYFIT XY;A;I;IFAIL;K;NEXTL;P;S;W;W1;W2;X;Y;⎕IO ⍝POLYNOMIAL FIT
  ⎕IO←1
  K←1+DEG
  (X Y)←XY
  W←(⍴X)⍴1                     ⍝ WEIGHTS
  W1←(3,⍴X)⍴0                  ⍝ REQUIRED WORK AREA
  W2←(2,K)⍴0                   ⍝ REQUIRED WORK AREA
  A←(2⍴K)⍴0                    ⍝ RESULTS AREA [1]
  S←K⍴0                        ⍝ RESULTS AREA [2]
  IFAIL←0                      ⍝ RESULTS AREA [3]
  (A S IFAIL)←E02ADF(⍴X)K K X Y W W1 W2 A S IFAIL
  A←A[;K]                      ⍝ REQUIRED COLUMN ONLY
  I←1
  Z←(⍴X)⍴0
  X←X XCAP⍳⍴X                  ⍝ NAG 'NORMALISATION'
  NEXTL←((⍴X)⍴NEXT),0
 NEXT:(P IFAIL)←0
  (P IFAIL)←E02AEF K A X[I]P IFAIL
  Z[I]←P
  →NEXTL[I←I+1]
****** **************************** BOTTOM OF DATA ****************************
```

APL86 Debate – 8th July 1986

APL Enhancements Don't Help the Spread of APL

Reported by Anthony Camacho

Chairman (Mel Chapman):

It is traditional, so far as the Olympics are concerned, that the host nation is entitled to nominate a new sport at each Olympiad. The APL86 Committee are therefore exercising that right and introducing into the conference proceedings DEBATES. I think it is the first time that such a concept has found its way into an APL conference.

The motion before this house is "APL enhancements do not help the spread of APL". Mr Richard Nabavi will speak first for the motion, and will be supported by Mrs. Valerie Lusmore. Speaking against the motion we have Mr Steve Schiavo supported by Mr John Myrna. I therefore call upon Mr Richard Nabavi to speak for the motion.

Richard Nabavi:

Mr Chairman, ladies and gentlemen, I must confess it was with very considerable trepidation that I accepted the invitation to speak in this debate on this side of the house in this particular venue. As I look around the people who are sitting in this room today, there are certain very well known faces in the APL world and I feel a bit like someone who has been invited to host a wine tasting at a meeting of the Temperance Society. What we are going to say on this side of the house may not be very popular with certain people here today, but I believe it is very important to the future of APL and indeed may determine whether APL has a long term future as a working computer language.

Now if we have a wonderful product, a really fantastic superb wonderful product, which no-one buys then there is something wrong. And the temptation at each successive APL conference over the last seven years has been to say "well the reason that no-one buys our product is because they are all stupid." You only have to look at some of the snide remarks that are made about other computing languages at successive APL talks (and I must confess I have made the same snide remarks) to see that our attitude to the outside world is one which can be characterised by the word "arrogant".

Let me just briefly say what we mean by enhanced APLs to avoid any confusion. We are not talking about file systems. We are not talking about having a decent error trapping primitive. We are not talking about good formatting routines. We are not talking in fact about all those good things that were done in the 70's by I.P.Sharp, S.T.S.C. and others. What we are talking about is what happened next which is the enhancements such as nested arrays, user defined operators, all those good things. In brief we are talking about APL2 and similar products from other vendors.

Now I want you just to imagine a brief scenario. We are going to talk about something different from APL. We are going to talk about a man who wants to buy a car. (I mean automobile for the American contingent!) So he goes into his local car showroom, speaks to the salesman and says he wants to buy a car. Now when that happens salesmen are normally very pleased because people often come in saying they don't want to buy anything. Someone who comes in saying they do want to buy something gets a really good reception. So the salesman says "Yes sir. What kind of car would you like?"

The man replies "I want a nice practical runabout just to go round town with the kids and the wife, do the shopping. I don't do many long distance trips. I want something with space so I can put luggage in it when we go on holiday; something cheap to run; something cheap to insure."

So the salesman says "Well we've got this fantastic Italian sports car. It is really amazing and very very powerful. It will do nought to sixty in 5.6 seconds and its top speed is 160 m.p.h. And it's really nice, look at it, isn't it beautiful, isn't it elegant!"

The customer says "Yes it is really beautiful, really elegant, I would really like to drive one of those some day. But actually what I am looking for is a nice runabout about town to go shopping with the kids."

So the salesman says "Electric windows – it's got the lot! It's got everything you want. Two seater, you can put a beautiful blonde next door and drive her off, and she will be really stunned by that!"

And the customer says "That's great, but how much petrol does it use? Can I afford to insure it? Where do I put the luggage? Where do I put the kids? Where do I put the wife when I've got the blonde in the front seat?"

So, not surpringly, the customer goes elsewhere and buys a boring standard old car. Now this story gets repeated again and again in car showrooms all round the country. Eventually the people who are trying to sell this product get the message – the customer is stupid! But they have got to react in some way so they go back to the R & D Department of the manufacturer and they explain the problem, saying "No-one is buying this car", so the R & D people say "Well we'll look at it" and they have got some really bright people, some of them sitting in this room. And they go back and they spend two years doing intensive R & D. They come back two years later, get all the sales people together at a big conference, maybe in Manchester, maybe in Seattle, maybe in Helsinki, and they say "We've got it now – this one is really going to go. This one is going to answer all the customers' objections to our product. Top speed is 164 m.p.h. and it will do nought to sixty in 5.2 seconds. Isn't it powerful!"

Now it may seem that no-one is that dumb, but I believe that the APL world has been that dumb. You can hear it, in fact we have heard it today, people are talking about the old APL, about how much more powerful the new APL is. But the old APL is not an old APL, it's not old hat, it is the most fantastic, exciting, amazing concept that the computing world hasn't seen. It's a brand new concept for everyone out there. It isn't an old APL at all, it's a totally new unknown product which we are not selling to the outside world.

Now some of those things they say about APL we know are not true. (The customer is stupid!) But some of those things undoubtedly are true, and some of them are facts which cannot be denied. And what has been the APL world's response to those objections? They have made it even more powerful! But despite the fact that they have made it even more powerful, they have not made it any more accessible to the outside world. Now there is a theory – and let me emphasise that it's a theory which I don't subscribe to – that IBM is a great conspiracy machine that produces conspiracies that are hatched in dark rooms somewhere in the United States. And one might think at first glance that APL2 was a conspiracy to stop the spread of APL. Of course I don't subscribe to that theory, but you have to admit that the timing is very very suspicious. I mean just look at it: just as the ISO standard for APL becomes fully developed and fully accepted, after years of work, IBM say "Yes but we are now going to move people from VS APL to APL2" which although it in theory conforms more or less to the ISO standard is such a radical extension that in effect it's a new language.

And look at the timing in terms of hardware. It may be coincidence, no doubt it is, but just as APL becomes available on small cheap micros that schools can afford, we change the ground rules so that APL2 and its lookalikes are not available on the small cheap micros that schools can afford. Because, make no mistake about it, the facts used to be that you could take an APL program, develop it on a small machine, a PC, even a Sinclair QL, and with very few exceptions that identical code would work on a mini computer, on a large super-micro, and on one of those big blue main-frames; and vice-versa. We have had customers in the company which I work for taking VS APL code and running it on a £150 micro computer.

Now it's not just an inconvenience that you can't do that with an APL2 system that uses APL2 features, it's an absolute disaster! The reason it's an absolute disaster is that you cannot really go to someone and say, "We have this wonderful product called APL. Here, try it on this cheap micro." And when they have tried it and got to like it turn round and say, "Well actually we didn't really mean that. That's old hat, that's ISO APL. What you really need is this wonderful APL2 product that can only run on a big mainframe."

All you have to do is to return to the terms of this motion and just think what is actually happening in the real world. We, on this side of the house, are saying that the enhancements to APL, typified by APL2 but also on the STSC and I.P.Sharp interpreters, are not spreading the word about APL. Now that's a pretty gloomy statement. I think it's true however, and if we want a ray of hope, let's look and see what is spreading the word about APL. Well I'll tell you. APL*PLUS PC has probably done more to spread the word about APL than anything else which has been discussed in these conference sessions. QL APL on small machines, other PC APLs, other cheap inexpensive micro APLs that are coming out – they are the vehicles which are going to spread the word about APL if anything is. But APL2 and the large mainframe systems – all that's happening there is that customers are migrating from VS APL to APL2. We are not getting new customers from that process; we are merely confusing the old ones.

So Mr Chairman, I would submit that we need to look at the real world, see what actually happening out there, and address the needs of the market, not what we think they ought to want.

Chairman: I now call upon Mr Steve Schiavo to oppose the motion.

Steve Schiavo:

I enjoyed my learned opponent's point of view and took a number of interesting points from it. I believe it is important that we clearly define what we mean by enhancements and what is the base upon which we are building enhancements. I believe that's made even more important, but if we can extend Richard's analogy about the automobile (the car for those of you on this side of the ocean), the runabout he described that the chap was trying to find down at the local car dealer did exist in a number of cases and does exist in some cases today, in some of the terms that he described it. But the kind of enhancements we propose and kind of things that his prospective auto manufacturer was installing are two different things. The kind of enhancements which we have installed in our line of business . . . are not simply the things to make his car go 170 m.p.h. They're to overcome certain inconveniences and offer additional conveniences beyond those described in the little runabout. For example some of the inconveniences in this little runabout are that its windows don't open, the fact that it has a roomy back seat but there is no way to get in and out of it. More importantly the little runabout that is available now, and has been for a long long time, gets about two miles per gallon. Now one might presume to take it upon oneself to improve that and actually tinker with the engine a bit but when you attempt to do so you find that the hood is welded shut.

All of which I believe leads us to clearly understand amongst ourselves, what do we mean by the enhancements. To start with, what is the base from which we are enhancing? I believe the base we talk about depends on what period of time we are talking about. It began some time ago with something called APL\360. For most of the 70s, and the 80s so far, it was a base of VS APL. In the 90s, possibly APL2, possibly APL PLUS, possibly a combination of one or the other or both. The enhancements that I'm describing I will limit to . . . modifications to the interpreter that add function for the APL user. I think the things we are talking about do include the file systems of various vendors, such things as CALL AP that extend also the environment and scope for APL users, SQL DS interface of APL2 and others, Sharp's packages, various . . . tasks that include interfaces to the rest of the world – I consider those to be enhancements.

For the purpose of this discussion what I will not consider to be enhancements (and you may argue but I wish you would defer it until I've left the country) I don't consider ports to different computer hardware with no additional function to be enhancements – it's just taking the same environment and moving it around. It's certainly of value to the market place, Lord knows it's been of value to us, but it's not an enhancement to the language. I certainly don't consider additional functions written in the language itself to be enhancements; those are merely conveniences which the customer could himself have written.

And finally what do we mean by the premise: have we spread the market, are more people using it. I think it's important that we are able to take credit for a net improvement in the market over what it would have been if there had been no enhancements.

We see the world in four classes of people, if you would visualise a little two by two matrix: those who know APL and use it; those who know APL and don't use it; those who don't know APL and don't use it; and a growing class of people who don't know APL and do use it – the way I describe the premise sounds strange but I do seriously include in this class the vast number of people who use applications created in APL and neither know nor care that it is written in APL in the first place. And that number is definitely increasing. I think all we need to do to demonstrate our side of the case is to argue whether movement amongst that two by two array is in our favour or to our detriment.

For example are there more people who know and use APL than there would be without a share file system, without a formatter? There are certainly more than there would be without a compiler – several of us in this room know of projects where APL would not be used today were it not for the availability of a dramatically less expensive more efficient user of CPU time.

Secondly are there more who use APL but don't know it than there would have been otherwise? Those are those who are using the applications and did nothing to write them and don't know what is inside them. Are naive users running APL applications today that would not have been written in APL without a fully featured error handler; without a secure multi-user file system; or without PC specific graphics and screen handling enhancements? You'll remember, to be precise, I don't consider simply moving the language to another piece of hardware an enhancement, but certainly a language that takes unique advantage of the additional features of that new hardware could be construed as an enhancement.

Let's talk about a net improvement to be gained by reducing the rate at which people exit APL: have fewer of those who know APL and not use it, have given up on using it. Fewer sites or applications have moved out of APL than would have done so without enhancements, I believe. I know of at least one firm which extended the life of its APL installation only because they were able to reduce the burden on the system by the use of certain important enhancements. And there are many sites who have fled the high cost of their APL timesharing applications, intending to re-write them in-house in some conventional language, only to re-write them on micro-computers in any of the wide number of enhanced APLs now available, with additional user-friendly features such as screens, graphics and so forth and all at a fraction of the original cost.

Finally, have fewer applications migrated away from APL because of the enhancements? Who could argue that enhancements have done nothing to persuade those using functional APL applications to avoid the costly conversion of their old systems? In fact their numbers are increasing even as we speak. STSC has many affiliated re-sellers, and I am sure a number of the other companies represented in this room have, who create and sell commercial applications as a business based on APLs that did not exist a few years ago. And their competitive advantage, those resellers, is based upon the speed and efficiency and user-friendliness provided not by the base APL systems but by an enhanced APL.

Meanwhile, more vulgar languages of the common people adapted to the day to day usage. They added an "alspufen" here and a "tickety-boo" there until they had enriched the language to an extent that they could make use of it everyday in everything that they did. It was no longer consigned to a small corner of their life for describing obscure species of plant life. It was no longer used as a way to set themselves apart in an esoteric conflab of like minded academics. It was used to conduct their daily work; to conduct their daily lives, and it expanded across what is now the whole industrial developed world. And that's what we would like to see come of meetings of groups like this; that's what we would like to see come of enhancements and continuing enhancements and continuing diversity and continuing arguments. Sometimes it's useful to discuss how many angels can dance on the head of a pin, although I think we would be much better served if we were to gather our strength and begin to tell the rest of the world why we have committed so much of our own resources and attentions to this language. Mr Chairman thank you. That concludes my point.

Chairman: Thank you. I now call upon Mrs. Valerie Lusmore to speak second for the motion.

Valerie Lusmore:

I find it quite ironic in a way that I suppose if I was still working for my first employers in APL, I would probably be sitting on the other side of the house, because I worked for many years for Sharp's in the stages when they were doing what I would call the original enhancements to APL. I came in at a base level where APL had what I would call the two basic and original enhancements, the formatter and the file system, which we would perhaps not consider enhancements but necessities. Without those I think APL would never have spread at all. Without those it would have been so difficult to do anything that nothing would ever have happened. Have I gone to the other side? No. ... I left Sharps. I went out in the real world. And, like many of us in this room, I thought APL was so marvellous that we wouldn't really need to sell it. That's a classic mistake we have all made. APL has spread despite the fact that most of us do not sell very well because we feel that something so obvious – yes, the customer is stupid! He must realise the advantages of this wonderful tool, after all if I find it a marvellous tool it's quite obvious the customer should discover it's a marvellous tool. I'm afraid the customer doesn't understand that at all – he says "Tough. I can buy this, that and the other, and I have a salesman dancing in attendance."

I think what would help is if we got our act together with the marketing; if we took things out there into the market place. In the times that we have sat here, all of us talking to ourselves, and we are still talking to ourselves, we haven't brought in very many people from the outside world to this conference. After all this is our private time away from the rest of the world. In that time, if we had put as much effort into spreading the word outside as we have put into improving things inside for ourselves, we could probably have doubled what is going on, and ensured actual continuity.

But the market place is outside and if we followed through (there's a very nice little phrase that I have heard in some of the literacy campaigns and I think we should adopt it for ourselves; and because it's APL we could each do it twice a year.) If we went into this and said "Each one, teach one" we'd be taking that out into the market place. We wouldn't be staying in here. We wouldn't be sitting here arguing about whether we can nest arrays or turn things upside down or whatever else. We would be going out and spreading it and I think that's what will spread APL, not us all staying here. And I would ask you to look at that Mr Chairman and say because it's APL "Each one of us, teach one twice a year. Each. Two rho." Thank you.

Chairman: May I now call upon Mr John Myrna to speak second against the motion.

John Myrna:
The automobile analogy of course is delightful. All of us, I'm sure, would love to be driving Model T Fords. There's the excitement of starting it every morning in the cold with the crank with your wife all bundled up in the car turning the choke. There's the thrill of excitement that comes from the knowledge that you're the mechanic, and if something breaks you're responsible for lifting the hood and fixing it. The fact that a trip to the edge of town is a thrill through the deep and endless mud; that there are no gauges to learn because of course there are no gauges; that there are an absolute minimum number of gears. But if we were to have continued manufacturing that kind of car as the road system improved, as the size of families changed, as the desire was to spread the use of that technology over an ever increasing number of people – people who were not necessarily interested in becoming mechanical engineers, road engineers, diagnosticians – the Model T would be – well I'll tell you what happened: Mr Ford got a little bit on that path in the 20s in which he said "You can have any colour you want as long as it's black", and while he was following on that purist Core Car Concept as he called it in those days, General Motors came out of nowhere, a consortium of people who just bothered to listen to what they were hearing their users ask for, and became a dominant force in the automobile industry. So I think the analogy is quite appropriate.

Now . . . . listen as I say let's talk about the real world. What is the real world? The real world is meeting the needs of the people who are using your technology.

Once you have learned the APL technology it's the easiest approach for you to use; why would you stop using it? Well you would stop using it because you need to produce pop-down menus, full screen applications, gain access to commercial databases, be able to run your applications in a production or batch environment. When you reach a point at which you can no longer do those things with your current technology, when you reach a point where you are forced to learn Lotus because the work-sheet organisation for entering and displaying data is the only way, the obvious way to do it, or you're forced out of it because you need just enough text editing that Del type editors are not satisfactory, then you will cross the barrier, you will invest, and you will learn something else. You'll learn a Lotus; you'll learn a DB2. The largest impact on keeping the level of APL usage up is on meeting the needs of the current users. And that means listening to the current users and enhancing the products, not based on what folks who are sitting back in Toronto, or sitting in Rockville, or sitting in San Jose are conceptualising, but in reacting to what the users are telling them they need to solve their problems.

The argument is that enhancements somehow inhibit the learning of APL. Well there is a simple APL. There is an ISO APL which is at the core of every single APL which is available. And those enhancements, all of them are layered. You don't have to know about complex numbers in order to become a beginner APL user. The idea that enhancements are available doesn't inhibit the decision to invest. It enhances it because when you sit down and you say "I'm going to invest so many hundreds of my hours in learning a new technology", you look forward to what is being done with that technology to convince yourself that there is value in making the investment. Are you keeping up with the new needs in the market place? Are you supporting the new hardware? As the needs change, is the technology going to change so that the investment I make today will continue to pay dividends in the future.

If you freeze things at the level of a Model T Ford, that's fine if you say that you're always going to live in the wilds on a muddy farm. But if you're looking to move to the big city in the future, you've got to be investing in a kind of technology that is going to move with with the times, is going to be driven by the needs of the user. Thank you very much Mr Chairman.

Chairman:

> *Thank you. I now wish to throw the motion open for discussion to the house. I will take points that are short and relevant from anyone who wishes to raise their hand and catch my eye. Mr Camacho.*

Anthony Camacho:
I don't believe either of them! It seems to me that the motivation is what you have to look at before you discover the real reason for enhancements to APL. And the motivation, it seems to me again, may be OK, fine, you've got customers who say "you've got to do this for me or else I'll go away" fine, you add something. But that's only half of it. The other half of it is to have a product that the others haven't got. And as soon as you split the products so that the others haven't got it, you have the problem that was raised at this morning's session, that the schools who are going to teach APL are faced with Product A and Product B, and they can't choose so they keep their hands in their pockets and they don't have an APL. And it seems to me that the people opposing this motion just haven't addressed that question. And I would like to hear them address it in their summing up.

Jim Lucas:
My question is with this debate. Both sides defined the scope of enhancements, and it seemed to me that the definitions were mutually exclusive. I don't think they have argued against each other at all. I don't know what they have argued against.

George Schlereth:
I was very much impressed by the idea that each of us should teach one ("Each one teach one"). I have been trying since 1975 to teach a lot of people APL; and in that time there have been enhancements in hardware, there have been enhancements in the APL offered. But one thing, in order to teach somebody, you must not make it complicated. Not everyone is so sporty that they want to become a mechanic before driving the car, so the approach I have taken is the approach of the standard of living. I ask them "If you want to drive a car, why do you want to become an automotive engineer?" Now, in that context, I found that it was an absolute necessity to have a screen editor, and to Hell with the Del editor. . .

Howard Peelle:
I would like to underscore the point made by Mr Camacho by saying that I found it surprising, if not reprehensible, that neither the pro nor the con positions mentioned the use of APL in schools. And schools, or the educational system, the formal system as we know it, comprises a very important part of the real world. Therefore I would like to ask if the debaters might address the question, in particular with regard to the introduction of APL notation to elementary school children.

Bernard Barnett (?):
I have heard one set of speakers today saying they want a language like this. I've heard another set of speakers today saying they want a language like that. It's the packages we are all selling. Who cares about the language. One of the previous speakers, Mr Chairman, from the floor, started by saying "I don't believe either of you." Let me finish by saying: Who the hell cares?

Malcolm Hawkes:
The number of people at the bottom, like myself, is massive. As we go up the pyramid we find the numbers decreasing. So obviously the sales of equipment etc. are greatest at the bottom where the equipment is the simplest. There should be all types of APL available from the most basic to the most advanced. The greatest sales will obviously be from the basic APL, where there are the most potential users. Those that want the most advanced features will be the ones at the top where there are fewer of them.

Neil Mitchison:
What I wonder is what we are really arguing about, it's a question of what we mean by the spread of APL. And I ask you just your emotional reaction to the news that some new package written in APL has sold ten thousand copies and none of the people who use it are aware that it's written in APL. Has that helped the spread of APL or hasn't it? What's your emotional reaction? I think it has, but my emotional reaction is to say "That's not what APL is about. That's not why I'm interested in APL."

Jonny Osterman:
I think you should sell, as some of you have said, solutions, or you should sell smaller parts of APL, not all the big things from the beginning. This has nothing to do with the enhancements.

Bob Sanderson (?):
I've come to the wrong meeting! I haven't got an APL. I've never used quad. Except for about two weeks I've never used APL in the sense that you use it, because I discovered an implementation of Iverson notation that works fine ... on my son's QL in November of last year when I was looking at an educational problem. I agree with both sides. Richard you definitely have to change the name of your package. I really do wish you would enhance it to do some of the things I need.

James Wheeler:
APL is today in use in artificial intelligence research labs in the United States by people who have at their disposal Smalltalk, Lisp, Prolog, the other buzzwords of the current leading edge of software technology, nonetheless have elected to use an enhanced APL for their software prototyping and for research into this area. And it is because APL has been enhanced in ways that allow them to represent the class of problems they're seeking to solve that, they adopted APL. They are new users of APL

Jim Lucas:
I have not heard of any company that has gone away from APL because of APL2. Almost all of the shops that I've heard of have converted to APL2 from VS APL very enthusiastically. And while I don't know that any new corporations have gone into APL because of APL2, the input that I have gotten from the people working in these other places is that the user base has expanded considerably as a result of APL2.

Mr Holmes (?):
Now I'm a bit mystified by this whole debate. But it seems to me that the most important part of the subject of this debate is not so much the enhancement as the spread of APL. Now if we look at the original book, the original intention of the APL notation as I believe it was, the original intention was to reform mathematical notation. Now if we can only get back to that grand idea we would see that we are falling down miserably because the sort of spread you people are talking about is nix, is nothing. What we have got to do, if we are going to realise what we all in our hearts believe should happen, is to kill off that old bloody notation that has been doing us such a disservice all these years.

Adrian Smith:
Firstly, we ought to look very hard at Pascal, which was solely an academic language until about a year ago, when someone sold it for fifty quid. That's the way you spread a language.

????:
In the last five years, the APLs have halved in price at the same time as the enhancements were taking place – so there has been a phenomenal increase in price-performance. The second issue I would like to point out is that I think, at the time of the Model T Ford, Ford was probably asking the same question: "Why are people not buying automobiles rather than buying horse drawn carriages?" And the point is the shift was taking place, and I think that is what's happening in APL. We are just not recognising it. In my country I train between fifty and a hundred people in APL every year. The number is increasing year by year. I simply think more and more people are moving towards APL precisely because of the enhancements and the reduction in price which has taken place.

Ed Myers:
I learned APL and I didn't even know what a matrix inversion was. Same thing in educational systems. There's no reason you have to bring someone in, settle down at the table and say "I've got five thousand eight hundred and twenty three different squiggles we could use."

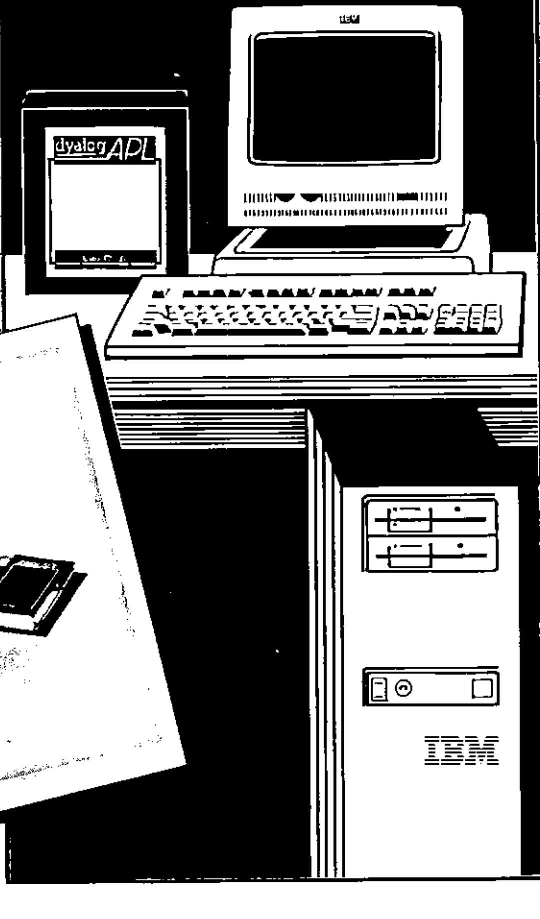
The English language is the same thing. I can get a dictionary out that has 50,283 words in it. Do you teach someone spelling by telling them to learn every word in that dictionary? Or do you teach them to read by explaining every single word in that dictionary? No, of course not. You explain that "Dick sees Spot. Spot runs fast." You learn things by building upon things, and you want to have languages that get things built.

This is all a bunch of rubbish.

dyalog APL

IBM 6150

IBM ANNOUNCES DYAL



IBM's announcement

notice for Dyalog APL

▼

◀ IBM 6150 Model 125

4.5 MIPS

High Speed Floating Point (1650 Kwip

High Performance disk (1.08 Mb/sec)

Dyadic is pleased to announce another distributor for Dyalog APL

Dyalog APL is a second generation APL that includes nested arrays, defined operators and other APL2 extensions. The 6150 implementation also provides support for standard IBM consoles, terminals and printers, graphics facilities and an interface to SQL/6150.

OG APL FOR THE 6150

IBM UK outlets for **Dyalog APL**

Over 66 authorised IBM 6150 dealers and your local IBM branch.

Announced as a **endor Logo** product by:

IBM Austria	IBM Netherlands
IBM Belgium	IBM Norway
IBM Denmark	IBM Portugal
IBM Finland	IBM Saudi Arabia
IBM France	IBM Spain
IBM Germany	IBM Sweden
IBM Iceland	IBM Switzerland
IBM Ireland	IBM Turkey
IBM Israel	IBM UK

Find out why **IBM** chose **Dyalog APL**

dyalog APL

Sales Department
Dyadic Systems Limited
Park House, The High Street, Alton, Hampshire, GU34 1EN, UK
Tel: (0420) 87024

IBM IS A TRADEMARK OF INTERNATIONAL BUSINESS MACHINES CORPORATION

Chairman:

I'm sorry I must now draw this debate to a close and I therefore call upon Mr Steve Schiavo to sum up for the opposition.

Steve Schiavo:

Before I get into my actual summary notes I do want to respond directly to some of the questions. Number one: where is the so called bicycle? Where is the simple version with the major features stripped out so that we can get back to basics? Well it does exist. I know of at least one version – it's called the Pocket. It is of the order of ten per-cent of the cost of the full system. It is available primarily not to people saying are we pursuing this as a profit oriented thing and not taking the long term about how we develop the market. It is a distinctly unprofitable product designed with the expectation that that would be the case because it is provided at ridiculously low price to educational institutions all over the world and through our dealer network. The point of it is to provide a starting base, not because we believe the availability of the enhancements makes it difficult to learn the unenhanced features of the language, but because we can therefore sell copies at a price that is attractive to schools.

Seriously, the enhancements to APL have brought the language out of the strictly academic and scientific community, and into regular use in many sectors of the business community. It is fast becoming the language of choice among actuaries and Wall Street analysts who are hardly likely to have so widely adopted it without even the most fundamental enhancements.

APL Plus, for one, is today used in the academic side of nearly five hundred colleges and universities. The important thing to remember is that the program has only been in existence for two years. I challenge anyone to say that that has not contributed to the spread. And it's used in those colleges and universities among not only the engineering and mathematics students, but also among students of finance, psychology, social sciences and business administration. It is no longer a purist's reflection of a clever mathematical symbolism.

And finally, and I think this is very important to all of us here in ways that may not be immediately obvious, the need for enhancements, the suggestion of a need for enhancements, has spawned an industry. S.T.S.C., I.P.Sharp, Dyalog, Analogic, MicroAPL, I.S.I., Portable Software, etc., etc. These firms have taught APL to thousands of paying customers. We have employed and nurtured hundreds of practitioners of the art who have in turn gone abroad in industry as enthusiastic emissaries of APL. They and you have used APL to create quality solutions for industry and government around the world, demonstrating the need, flexibility, and function of APL features which are not in any standard and without which the APL community would be a barren and lonely place even today. Thank you, Mr Chairman, thank you.

Chairman:

I now call upon Mr Richard Nabavi to sum up for the proposition.

Richard Nabavi:

Thank you Mr Chairman. Thank you ladies and gentlemen for your interesting comments. I am actually quite surprised, at a conference of this kind, to find so much agreement with what I have been saying. Agreement even from people who think they don't agree with me. I just wish I hadn't brought up that analogy about the automobile. Next time I won't bring up any analogies.

I would like to take up two points from people who I think oppose this motion. One was made by the honourable speaker over here who I presume opposed the motion. Another by Ed Myers who I understand also opposes the motion. The strange thing is that both of them gave very cogent arguments for supporting this motion. Now the problem with a debate like this is that people get confused over what we are talking about. So let me re-state what we are saying. We are saying that enhancements to APL, by which we mean enhancements to the APL language, are not doing anything to help spread APL. And, as the honourable gentleman over here said, the world outside, the non-APLers, they don't care about strand notation and bracket notation and rank operators. It's precisely what he said. Exactly. They don't care. It doesn't matter. The fact that we get those right or get them wrong or don't do them or do do them will not help to spread the use of APL. Precisely.

Ed Myers' point was even more cogent. Except that it was cogent on the opposite side to what he thought. Because what Ed Myers said was that when he was learning APL he didn't know or care about the matrix divide operation (the matrix divide primitive). Precisely. If that enhancement if you like, or that feature, had not been there he would still have been interested in APL. It was not an issue. This is exactly what we are saying. The reasons that people do not use APL are not to do with lack of feature. We all know that APL, even the simplest, even the most basic APL, has got much much more feature than any other computing language. Adding more feature may or may not be a worthwile thing to do. We are not saying it is wrong necessarily. What we are saying is what this motion says: that it does not address the point, which is to spread the use of APL.

Now there has been a diversionary tactic used by the opposition in this debate. They have talked about full screen editors. They have talked about file systems. They have talked about graphics interfaces on the PC. And we wholeheartedly support such features. They are absolutely essential if APL is to be widely used. But let's not get confused, those are not enhancements to the APL language. They are environment specific features which you add either as defined functions or possibly as quad functions. But they don't affect the APL language itself. They don't address the point which is being made which is that the enhancements to the APL language are not needed. Yes, of course we need all those things. We need the windows. We need the icons. We need the full screen editors. Wonderful. Let's go on and write those, deliver those to the customers. But let's not argue about what notation we use for nested arrays.

There is one very interesting point which I think is one which is perhaps a little unpalatable. This is a point which several speakers have alluded to, and I think George Schlereth in particular mentioned it, which is the following. We have got to make APL less frightening, more accessible. Now that means that there is a cost associated with putting extra features into an interpreter, into the APL language. Clearly you can go on adding features ad infinitum, and indeed some APLs give the impression that they have gone on adding features ad infinitum. And people have been saying "Well that doesn't really matter. You don't have to use it. You don't have to worry whether it's there or not. Maybe I'll come across it one day but in the meantime it doesn't matter."

That is not true. The extra complications inevitably make it more difficult to teach people APL. They make the manuals look more frightening. They make the teaching courses longer. They make the product more expensive (someone has got to pay for all that R and D). They make the interpreters larger, so they can't fit on the small machines. There is a cost associated with every feature you add. And the question which we are asking here is: is that cost too high? When we ask that question we are asking it not from your point of view, not from the point of view of people who already know APL. We are asking if that cost is too high for the school kids, the universities, the small businesses, the large businesses which are tight on budgets. Are those extra features, which cost something, gaining us anything in helping to spread APL?

I keep coming back to this: you have got to get back to the real point about about the motion. Not are these things desirable, are they elegant, are they powerful, but do they help spread APL? Ask yourself this question: do you think it would be any easier to sell APL to someone who hasn't used it, who has never seen it before, as a result of extra language features that are put in? Or is that an irrelevance to the main problem which is how we get out there and sell the product. Mr Chairman, ladies and gentlemen thank you for your time. I hope that we have learned something from this debate. What I have been surprised by is the lack of anyone really saying that all these advanced language features make much difference. The only strong arguments we have heard have been on the interfaces, the environments, and there we agree with you. I think this motion should be carried unanimously. Thank you.

Chairman:

I now propose to put this motion to the vote. The vote will be taken by a show of hands.

* * *

I declare the motion carried. Thank you very much ladies and gentlemen.

The APL Entrepreneur

Panel discussion at APL86 in Manchester

reported by Val Lusmore

Introduction.

The participants were:

Stan Wilkinson, Chairman, is a founder of HMW Programming Consultants and, as such, has experience of the world of APL business himself. The large and enthusiastic audience was divided between those who might already consider themselves as APL business-people, and those who were wondering whether this might be an interesting option for them. Many were quite good at self-publicity, having found this a necessity for being in business, so Stan had quite a difficult task keeping them to the topic.

Henry Brudzewsky, from Denmark, has been an APL consultant for many years – his wife also works in the business, and they have wide experience in several countries. This 2-person consulting team is the typical size of many APL companies – so Henry was representing the majority of APL consultancies.

The other 3 panel members were from some of the largest APL consultancies in Europe, all of which started off very small. They all shared advice and experience for those wishing to follow the same route.

Romilly Cocking, of Cocking and Drury, the English consultancy established 10 years ago as a two-person operation, now has 35 staff – most of whom are consulting full-time in APL and related problem-solving tools. He also has the UK agency for APL*PLUS, and makes about 10% of his revenue from APL Education.

Valerie Lusmore, of APL People, started as a freelance consultant when she had her first baby in 1979. Two years later she joined up with two other like-minded spirits and now has a linked group of five companies all related to various vertical market-places where general APL problem solving has been used as a tool. They employ between 20 and 30 people, and operate an employment agency solely devoted to the APL community.

Timo Seppala, of TMT team in Finland, started 5 and a half years ago with some of his colleagues who had been with him at IBM – they now have 30 people doing APL consultancy in four Scandinavian countries, with annual revenue of £2.5 million. The group has people doing non-APL consultancy as well, but it was the APL side that started the business.

All the panel members gave some personal thoughts of how and why one might start up an APL business – a general consensus was that, at first, it was marvellous to get paid for doing things one enjoyed and that the service was much required. They all touched on the standard problems of being a consultant at a client – and pointed out that one has to have both the ability and the personality to do this and that many people do not enjoy it when they find that consultants and contractors are often resented by the permanent workforce. They also discussed the myth that one was much better paid as a contractor than as a permanent employee – the view is that one ends up just about as well off as someone on a good permanent salary with benefits, and that one should think very hard about the whole thing before throwing up a good job to go contracting.

Val, who spends much time counselling members of the APL community on their career plans, has also dealt with many people who have been made redundant overnight – a shattering experience for anyone, even if a reasonable golden handshake is offered. She finds that most of us in the APL community spend no time on planning our lives and careers. All of the panel agreed that you should look carefully at whether you are suited temperamentally to running your own business and what your options are.

The three who had experienced the growth from a very small company to a much larger one discussed the problems of growth, although all agreed that in general business terms their firms are all still tiny. Growth problems are chiefly those concerned with management, cash-flow, people (how to find them and how to keep them), the market-place you are in, and how to handle the changes. All agreed that finding work is always a problem, and handling the peaks and troughs is always difficult.

There was general consensus that the main objective was to increase the size of the market where APL could be used for problem-solving, and co-operating where possible, rather than allowing the customers to force them to compete. Romilly said they found themselves bidding now more for work against other non-APL companies, where their lower total cost should always win, but all said they had found they had to use their consulting skills and also to use other non-APL tools where appropriate – rather than being dogmatic. APL is seen as a tool for problem-solving that should give them the competitive edge over others rather than an end in itself.

Edited transcript of panel discussion

Stan: This session is called the APL entrepreneur. We have four people who have admitted to being such – Romilly, Timo, Henry and Val. They will introduce themselves before they start arguing.

Romilly: My name is Romilly Cocking of Cocking and Drury. We started up about 9.5 years ago as a 2-man business with 2 objectives – we were going to do something which was fun and we were going to do something which was profitable and we figured we could get away with both those things by offering an APL consultancy service. We have grown a bit since then and have continued to achieve both those objectives. There are three key secrets in the way that we have developed our business. The first is to find out as best you can what IBM is doing or planning on doing; the second is to try and understand why; and the third is to look at all the many opportunities that creates and pick the ones which are easiest and most profitable to attack. I think I probably speak for all of us – we owe our business to IBM who created a demand for APL services which in effect they chose not to provide themselves and thus gave an opportunity for other people to fill the vacuum created.

Henry: I started in 1972 – and as an example to illustrate my work I have borrowed a special shoe from my wife, which has the very rare property that it was shaped according to the human foot, not according to fashion. My work is to shape the system to fit human beings and not to cripple people into the existing systems.

I wrote down five things, you should remember if you start on your own. First you must be prepared to keep up to date – APL2, PCs, DOS, UNIX, VM, VSPC, CMS, and also hardware. Secondly, how do you organise the company? In Denmark you can either have a private company or be personally responsible to your customers. I think the customers trust you more if you are personally responsible and this is the way I do it. Thirdly, you must choose whether to be an agent for software or hardware or be truly independent. Whenever you make a study which involves a choice the one which loses will blame you even if it is the choice of your client. The fourth problem with being small is the changing workload – sometimes I work day and night and have very little time to sleep and then suddenly there is some delay and we have wonderful opportunities to take a vacation off-season. One way of getting around this is to cooperate with competitors. Lastly, you must be better than the others – do a first class job, give very good documentation to solve the problem of maintenance. Many of you have seen our baby son. He has not started walking yet, but we have decided to give him shoes that fit his feet so that he will be able to walk through life without being crippled.

Timo: Five and a half years ago I started a company with two of my colleagues from IBM. We didn't really have a clear idea of what we were going to do, but we thought like Romilly that it had to be fun. My advice is firstly to make a study of the market place. Is there one? Secondly, if you are a consultant, you can either consult or sell your consulting services but you can't do both. You may be living on social security for some time. Thirdly, remember that consulting is a very lonely job. You are usually in a customer's premises which can be a hostile environment. The staff like to have your work but you aren't really part of the firm. Lastly, what makes you think you are better than others in the marketplace – have you anything special? If so, go out and sell it.

Val: I started off on my own in 1979, and unlike the others I didn't decide I wanted a business. I had my first child and with 10 years' consulting experience in small companies on 3 continents I wasn't exactly a novice. However the powers that be didn't know how to cope with someone like me who didn't conform to the accepted working patterns – so it was easier to organise if I became my own boss. When we had timesharing it was easy to freelance and very enjoyable. After a couple of years I was joined by Dave Alis and Tim Perry and we've now got 5 associated APL companies. That sounds very glib but it all seemed to happen quite naturally. My advice would be similar to that given by all the others, but because we run an employment agency our perspective and experience of the job market is different and I think the audience may find this experience useful. What does this business of running an employment agency mean to our customers, and to the way we run our business?

It means we employ contractors half the time as our way of coping with overloads. Our initial aim was not to grow faster than we could afford. However, we found the advantage of using people with expertise in particular areas outweighed this. The extension of the business into finding permanent staff came in because of customer demand. This aspect means that I talk to many people working in APL about what they want to do and also to employers about their needs, both short- and long-term. We give advice, acting as a sounding board for people to think about planning their own career; to talk about growth and progression. All these are things that many of them last did when they were in school, but were too young to appreciate. We also have had many people recently, who are made redundant almost overnight (often after 20 years) who have never thought of themselves outside the context of that particular employer. Some give a golden handshake, but it's difficult to accept that it's difficult to move when you are 45, especially if you don't want to go into management.

Some people we try doing contract work and thus starting their own business. Some like it, others loathe it. That is only the start; you then decide if you're a loner or if you are happiest with somebody like me finding your next job or working for someone like Romilly or where you feel happy temperamentally. It may be possible to take that as a short term option and really look for the kind of job that suits you. Well we have had the lecture on jobs – let's go back to being an entrepreneur.

Romilly: Can I start with a question? The market has changed a lot since we all started up. It has got bigger and there are a lot more people in it. Do you think it is easier or harder now for someone starting up?

Val: I think it is harder. There is so much advice available to would-be business-people that you could be swamped in it. We were lucky in that we didn't always realise all the pitfalls – we just went ahead and did it.

Henry: Today it is far easier to start up. Why just 5 years ago, when you needed a terminal with the funny symbols we all know and nobody had that if they didn't use APL – you had to subscribe to a time-sharing service or you had to have APL in-house. Today you can bring one or two small diskettes to a future client. He already has the PC and then you can demonstrate what APL is able to do for him. You can even bring the latest package you did for another client (if your other client allows you, of course), and demonstrate it to your future client. Then he will get enthusiastic, especially if you tell him how cheap it is to develop such a system. I find it much much easier thanks to the PCs.

Timo: I don't think consulting in a PC environment can be profitable. People think anything on a PC costs less than £100 so that when you are quoting work for, let's say, a very special upper case that takes a month of your time and you get £100 for that so . . . .

Henry: Well I don't agree with Timo. I didn't say I was going to continue using the PC but I can demonstrate what APL can do on the PC and then I can sell APL to the customer and he can run it on his mainframe and we can continue with big business.

Romilly: There is presumably a simple solution. If consulting costs a lot more than the PC – why not sell the consulting and give the PC away free?

Val: So many people think they will get rich writing software – they don't understand the economics of development, marketing and maintenance. Unless you are extremely well organised, it's an easy way to lose a lot of money fast. It's like any other task that everybody knows how to do – remarkably simple in concept, just difficult in practice. Several times every year we get people who think they'll sit down with their PC and just develop a piece of software to sell – and make their fortune.

Question: Do runtime APLs help bring the cost of software down?

Valerie: If the volume is big enough, then the run-time APL interpreter is obviously useful – we have looked at one particular project where the projected volume would be tens of thousands of copies of the software – and for that we'd negotiate a very cheap rate for the interpreter as a part of the software. The problem is when you only sell 10 copies of a piece of software – you might well not cover the development costs.

Timo: I think it is a question of the market place you want to be in. Nowadays few companies buy our services because we are APL consultants – it is solutions they are buying and I have found that we do not promote APL as such very much. We are competing against other consulting houses that deliver applications using ICIS and Cobol and if we can make our proposal half of theirs we will still make a nice profit.

Romilly: Going back to Val's point about profit in consulting I think if you are selling people's time instead of solutions it's hard to make a profit. As a rough rule of thumb once you have fixed overheads you have salaries and rent to pay, you can lose in a month what you make in a year and that is something to consider before you take on staff.

Val: I think it's the only thing I agree with Mrs Thatcher on – I prefer not to spend money I haven't got. We set up very small and if we didn't have money coming in, we didn't spend it. We used contractors so we didn't spend money on people unless we had money coming in from clients. It is easy to lose money unless you have enough work coming in and that is the problem with being on your own. It's very easy to lose money in many classic ways by doing the wrong things. Look at the APL engineering company we bought last year. They had been around for years, they have a great product but were selling it at the wrong price and giving all the enhancements, which cost a lot to write, away free. We stopped that immediately. You can't afford to give the consulting work away – you must sell things at a price that is perceived as good value for the job. That is obvious if you stop and think about it.

Question: Are there still business opportunities?

Val: We are constantly looking at other business opportunities. We have to because the market place is surely not just APL and in order to survive we are all going to have to go out and grow that marketplace – I agree with Timo, we are selling solutions, not APL. Our main objective is to survive – and if anyone has a good sound proposition I'd be happy to look at it.

Question: How much of your work is purely APL and how much is APL working with other languages?

Romilly: About one job in three involves the use of either pre-existing software that isn't written in APL or coding specially developed. On the mainframe, if you have efficiency problems that cannot be solved neatly by APL, you may have to drop out of APL for a while. On the micro it tends to be again that you want to do something really close to the operating system so that you come out of APL for a while and then come back again. Now it's about one job in three, a year ago it was about one job in ten.

Val: How many of you are based outside Europe, because I suspect that the North American experience of running your own business is different to ours. People I meet seem to work in the US as independents. We in Europe are working in groups that have grown. We're not really in competition but joined together in pushing out the frontiers. Is it like that elsewhere?

Answer . Virtually every APL contract I've been on has other APL firms on the same contract, and this makes it harder.

At this point Ed Cherlin from APL Market News produced a survey of the leading APL consultancies in North America, most of which have one or two people. The largest is 8 consultants.

Romilly: Things have changed in the last two years – years ago, in nearly all the contracts we did they said 'We want an APL system and we want you to do it.' That was it. These days it's very rare indeed to be so lucky.

Val: We find ourselves bidding more against other APL companies. I notice that the customers try to force us into spurious competition.

Romilly: We find we are bidding more and more against non-APL companies who are offering solutions and the fact that you're using APL is not necessarily relevant to the contract as Timo says. What you're doing is simply going in at a lower total project cost and that should be why APL is going to win every time. We also tend to see people looking at an externally developed solution vs an internally developed 4GL solution.

Question : Who are your customers? The Informatics dept or the users?

Romilly: it splits about 50-50. For us customers are TIMES 1000 companies – they have at least a million pounds of hardware in their machine-room or they wouldn't be running APL.

Timo: What has been changing is that 10 years ago there were no 4GLs, no package programs, no PCs. What APL was used for were PC applications. That's why VM/CMS was so popular – it was a massive virtual PC. Now we have PCs and 4GL packages, that side of APL work is gone totally. They do it themselves with Focus, Ramis 2 etc. APL is now used for serious data processing. Serious means the application is difficult and the most profitable thing to do is to write applications that give the customer the competitive edge. If you find applications where the customer can make more profit, you can price them much higher.

Val: The aspect of providing added value is really important to our profitability. I'm constantly surprised at how much of our revenue comes from giving added value – project training, extra software, hardware, applications in other departments – this may be 10-20% of the contract.

Henry: This time I do agree with Timo. I was abroad for some years and my Health insurance was in Denmark. During a visit there I went to their offices to complain that I was underpaid. While I waited to talk to the manager I thought it over and saw it was quite difficult for the insurance company to pay the exact amount – I was in a country with 400% inflation and that's why I got too little. So I asked the manager if it was a very difficult problem with people travelling all over the world and he told me he had many long letters even if he only erred by a small amount and I asked him if he would like a solution, and he is one of my very good clients now. I was very lucky that I came at just the right moment.

Question: Valerie suggested that one should fund oneself from one's own resources and I'd like to ask the panel how much they had to borrow to get going.

Timo: When we started we cut our salaries to half what we had at IBM and we didn't have any summer vacation for two years. Then after a year the bank manager came to us and said 'Hey guys, don't you need any money? Don't you need a credit line?' When you get big then you need capital – if you want to buy an IBM 3090 that is really expensive. But you are making a big mistake if you start by borrowing a lot, you may well do no better than if you had no capital, and you also have the debt to pay off. Wait until you need money for capital items.

Val: I didn't get a penny for three months and then the cash started to come in, so I didn't start by borrowing – I think I lived on my maternity benefit. Now we are bigger we have borrowed for particular projects.

Romilly: When we started up we had enough money to live on for a few months with the help of a working wife, but since then we've typically needed to go up to 6 weeks' turnover in credit facilities.

Val: Six weeks is exactly what we find. Cash flow is very important – most of our customers are big companies who pay late, probably because of their complicated computer systems. You have to be prepared to go in right at the start and sort out how they pay and who is responsible for getting your bills paid, even though it is much more interesting to get on with solving their technical problems. If you don't, you will regret it.

Romilly: If you are expanding fast then the net negative cash-flow can get very high and when you take somebody on they expect to get paid from the day they start working for you – they may not fit into a job the day they join you and if you include the cost of recruitment (advertising, agency fees, interviewing) it may be 4 or 5 months before they make a positive contribution to cash flow. That 6 weeks figure is against a rate of growth of 70% per annum compound internally funded, but that's about the limit that way. If you go any faster you need (a) money and (b) management.

Val: It's the management people that are most difficult to find, the money is relatively easy once you are growing and have a track record. You also have to make a personal decision whether you go on being a technician or whether you enjoy running a business – and if you don't enjoy that change, recognise this and organise to stay on the technical side. You must work as hard at finding the new skills as you once did acquiring APL expertise.

Romilly: If you grow at that rate the fundamental problems your business is facing are changing over a two-year horizon. You start worrying about structure, long-term planning, and I don't know what comes next.

Question: What are the implications of having employees? Outside the USA – where they come and go relatively freely.

Romilly: If someone has worked for you for about a year then it's not possible to get rid of them without having either a very good reason or because they are fundamentally and demonstrably unsatisfactory or because you can show that the job they were doing has gone away and you then have to make some sort of compensating payment.

Val: One of the main problems everyone has is finding enough skilled staff. I did study the cost of our setting up a training scheme just for APL – taking on juniors, teaching them APL and then teaching them to become consultants. If they didn't learn APL you could tell quite quickly, without too much loss on either side, and if they weren't going to make the grade as consultants you could tell within a few months and we could sell them on to customers as trained programmers. I found we would need to take on about a dozen at a time to not lose money – the investment per person was two or three thousand pounds, and there is a fairly high level of risk – so we've put that on the back burner for now.

Timo: The other side of the problem is how you keep your people. In a consulting company the only real capital you have is people and your reputation. Our policy is that after you have been with TMT for 2 years you can get shares in the company – most of our people are shareholders. Everyone wants to be a capitalist.

Val: I think our business will never grow unless all the people working for it feel that it's their business and contribute to that growth, and that the profits are not just going to the people who started it up. They've got to feel they are vital to the success of the company – but they all have to be salesmen too because the best salesman you've got are all those consultants out there – doing a good job.

Question: How many people are in APL Education?

Romilly: About 10% of our revenue – quite significant.

Timo: You have to train your customers to buy from you.

Henry: Giving a course in APL usually gives you one or two good clients. Also at universities and schools you may get very interesting business.

Val: If we don't teach the APL community stays static – and besides survival I would see our main task as co-operating in enlarging and improving that market place.

Stan: On that note, I must thank the panel – they have given us a lot to think about. As you have heard there are many opportunities to make a living out of APL – but you must be prepared to grasp every one!

APL CONSULTANCY

* Debugging
* Support and maintenance
* Feasibility studies
* Interfaces to mainframe systems
* Conversions
* Major developments

Our highly experienced consultants are capable of tackling any APL problem large or small.

For further details call Valerie Lusmore on 0225 62602

APL People Ltd
17 Barton Street, Bath BA1 1HQ
Tel: 0225 62602

Introduction to General Articles

Our General Articles start this issue with the latest extract from Anthony Camacho's series of "Steps to a Better BASIC". These articles were designed to introduce the concepts of APL to an audience familiar with the conventional BASIC language, and were originally published in *Datalink* magazine, with whose kind permission they are re-printed.

Dick Bowman, the Chairman of the British APL Association, has written explaining the thinking behind an APL prize scheme to reward those who have contributed most to extending the publicity of APL to the world outside the APL community.

Robert Pullman is director of The Rochester Group, a New York based consulting company. He has sent us some ROTs (Rules-Of-Thumb) on APL system design. 'The Programmer as Designer, the Designer as Detective' is a valuable but light-hearted look at this problem.

Our final contribution comes from Sylvia Camacho, who looks with the benefit of hindsight at developments in computing and the role therein of a mathematical notation

Steps to a better BASIC

by Anthony Camacho

Each with each: the table maker

In the introductory article to this series, I suggested that a useful innovation in BASIC would be the command EWE(*) which could take any symbol in the brackets and would combine each of the members of one array with each of the members of the other using the chosen mathematical or logical symbol between them.

APL has just that function. It is called the outer product and it is written as the small circle, the full stop and the symbol for the operation required. If in L we have the integers 1 to 12, here is "L jot dot times L"

```
      L∘.×L
 1   2   3   4   5   6   7   8   9  10  11  12
 2   4   6   8  10  12  14  16  18  20  22  24
 3   6   9  12  15  18  21  24  27  30  33  36
 4   8  12  16  20  24  28  32  36  40  44  48
 5  10  15  20  25  30  35  40  45  50  55  60
 6  12  18  24  30  36  42  48  54  60  66  72
 7  14  21  28  35  42  49  56  63  70  77  84
 8  16  24  32  40  48  56  64  72  80  88  96
 9  18  27  36  45  54  63  72  81  90  99 108
10  20  30  40  50  60  70  80  90 100 110 120
11  22  33  44  55  66  77  88  99 110 121 132
12  24  36  48  60  72  84  96 108 120 132 144
```

It makes things look rather better if we embed this in a defined function which also prints the values of the arguments along the top and left side, and puts the symbol being tabulated in the corner of the table.

Here is a function that does all that:

```
    ∇ R←FN TABLE VEC
[1] ⍝ PRODUCES TABLE OF VEC∘.'FN' VEC
[2] ⍎'R←VEC∘.',FN,'VEC'
[3] R←⍕(0,0,VEC)⍪0⍪(((⍴VEC),1)⍴VEC),0,R
[4] R[2;]←(¯1↑⍴R)⍴'-'
[5] R[1;R[1;]⍳'0']←FN
[6] R[;R[1;]⍳'0']←(1↑⍴R)⍴'|'
    ∇
```

The header shows that it gives an explicit result (we could store the result in a variable if we chose), and that it takes two arguments: the left argument is the function symbol and the right argument is a vector or a row of numbers. Line 1 is just a comment. Line 2 calculates the table; the first character 'executes' the string of characters following as an instruction – that is how it is possible to give a choice of symbol to make the table from. Line 3 puts the variable values down the left-hand side and across the top, with a row of zeroes in between them and the main table. Line 4 replaces the line of zeroes with a line of hyphens. Line 5 puts the symbol being tabulated in the top left corner and line 6 replaces the column of zeroes with a column of vertical bars.

Here are some results. A division table up to 5 (with the print precision reduced to show only four places of decimals – APL can show up to sixteen):

```
      '÷'TABLE ⍳5
÷       |     1      2      3      4      5
--------|-----------------------------------------
1       |     1      0.5    0.3333 0.25   0.2
2       |     2      1      0.6667 0.5    0.4
3       |     3      1.5    1      0.75   0.6
4       |     4      2      1.333  1      0.8
5       |     5      2.5    1.667  1.25   1
```

A table of powers up to 6:

```
      '*'TABLE ⍳6
    *      |     1     2     3     4     5     6
-----------|------------------------------------
    1      |     1     1     1     1     1     1
    2      |     2     4     8    16    32    64
    3      |     3     9    27    81   243   729
    4      |     4    16    64   256  1024  4096
    5      |     5    25   125   625  3125 15630
    6      |     6    36   216  1296  7776 46660
```

And here is one of the more interesting functions – combinations – which gives Pascal's triangle:

```
      '!'TABLE 0,⍳9
   !   |   0   1   2   3   4   5   6   7   8   9
-------|----------------------------------------
   0   |   1   1   1   1   1   1   1   1   1   1
   1   |   0   1   2   3   4   5   6   7   8   9
   2   |   0   0   1   3   6  10  15  21  28  36
   3   |   0   0   0   1   4  10  20  35  56  84
   4   |   0   0   0   0   1   5  15  35  70 126
   5   |   0   0   0   0   0   1   6  21  56 126
   6   |   0   0   0   0   0   0   1   7  28  84
   7   |   0   0   0   0   0   0   0   1   8  36
   8   |   0   0   0   0   0   0   0   0   1   9
   9   |   0   0   0   0   0   0   0   0   0   1
```

And of course you can get Boolean results too. Here is the greater-than-or-equal-to function:

```
      '≥' TABLE L
  ≥  |  1  2  3  4  5  6  7  8  9 10 11 12
-----|------------------------------------
  1  |  1  0  0  0  0  0  0  0  0  0  0  0
  2  |  1  1  0  0  0  0  0  0  0  0  0  0
  3  |  1  1  1  0  0  0  0  0  0  0  0  0
  4  |  1  1  1  1  0  0  0  0  0  0  0  0
  5  |  1  1  1  1  1  0  0  0  0  0  0  0
  6  |  1  1  1  1  1  1  0  0  0  0  0  0
  7  |  1  1  1  1  1  1  1  0  0  0  0  0
  8  |  1  1  1  1  1  1  1  1  0  0  0  0
  9  |  1  1  1  1  1  1  1  1  1  0  0  0
 10  |  1  1  1  1  1  1  1  1  1  1  0  0
 11  |  1  1  1  1  1  1  1  1  1  1  1  0
 12  |  1  1  1  1  1  1  1  1  1  1  1  1
```

Coming out of the closet: taking the Message to the People

by Dick Bowman

In recent months the B.A.A. Committee has increasingly been thinking about the way the APL community meets its own needs for communication about APL and the perception of APL which is held in the rest of the computing industry. Our conclusion is that there's a gulf; we now want to see if there are ways in which we can help bridge the gap. As is our wont, we want to do this in a way which involves a lot of other people in doing the hard work – but bear with us because we also offer rewards.

The way we see it is as follows. The APL community is doing an absolutely splendid job of talking to itself; journals like Quote-Quad and some of the other national group publications are essential reading for the practising APLer, and we have an excellent annual conference which bursts at the seams with material we can use. But if you're not already within the APL community, you don't know about this activity, and even if you did know it's unlikely that you would find it compulsively attractive.

What the rest of the world knows about APL is culled from the general computing press and the various conferences and seminars which take place. The press is voicing opinions like "APL is a language that looked promising in the 70's but never really made it", or it has short introductory pieces from people who have just found out about APL (and often seem to have grasped the wrong end of the stick). APL is very rarely mentioned at major computing events; we hear rumours about people submitting papers describing APL-based work which get rejected purely because APL isn't one of the 'officially-approved' languages.

One thing which APL does have going for it is that most of the computing world knows of us; they may have weird notions, but at least we have some awareness (even if it's the wrong sort). We have notoriety, let's change it into respect. The current view of the B.A.A. Committee is that we should encourage people to try to present papers at events outside the APL circle and to write articles for the non-APL press; the people who are really using APL effectively should share their experiences and knowledge. What we've done so far is to create a starter list of events and publications which we think would be improved by contributions from the APL world. This is only a starter list and we will be overjoyed by anyone pointing out the virtues of the ten zillion places we haven't included. As it stands we have (in no implied order):

Events

Siggraph Annual Conference (ACM)
OR Society Annual Conference BCS Human-Computer Interaction Group Conference (annually, September)
BCS Expert Systems Group Conference (annually, December)
Royal Statistical Society events
BCS Software Engineering Group Conference (annually, September)

BCS Medical Group events
BCS Computer Education Group events
BCS Computers in Education Group events
Independent Schools Microelectronics Centre

Publications

BCS Expert Systems Group Journal
Computing
BCS Computer Bulletin
Computer Weekly
BCS Software Engineering Group Journal
Datalink
Mass-market 'personal computing' magazines
Practical Computing
OR Society Journal
Byte
BCS Computers in Education Group Journal
BCS Human-Computer Interaction Group Journal
BCS Medical Group Journal
Data Processing

Note how the British Computer Society figures in both lists; we are a Specialist Group and one of the benefits that this brings is that we have somewhat easier access to a vast range of computing interests, events and publications. We want to exploit our advantages.

Topics

Obviously suitable material for one place is not necessarily attractive to others; some events or publications may have restrictions which do not apply to others; and some routes may already be heavily over-subscribed.

The general themes which we, innocently, think are most sensible to expound on are biased heavily toward applications – the why and how of using APL effectively; the benefits to an organisation of doing certain things in APL; how to choose whether APL is a suitable vehicle. Essentially it seems good to emphasise novelty (give them a new viewpoint on things they believed they knew all about) and lace it with a little controversy (even if only in the title).

What we think needs emphasis is the effectiveness of APL as a problem-solving tool, and the fact that it is a tool which is relied on as a means of delivery in many organisations. APL is already in place, it's not a solution looking for a problem.

Remember that just because YOU are the person doing it, something need not be boring. You almost certainly have things to say and to write about which are worthy of a public airing. The B.A.A. is more than happy to offer pre-publication counselling; if you have an idea which you think could go further and you want to discuss ways of advancing it, then contact any of the B.A.A. Committee.

We are also interested in collecting information after the event – reviewing how it went on the night. Even if the unthinkable happens and your efforts are rejected, we would like to know why. It helps us understand the world's perception of APL, and other people to avoid making the same mistakes.

Sticks

The 'mainstream' computing world is capable of totally ignoring us; some factions within that world might even see this as a positive strategy. If we want APL to grow and be used in those places where it can most effectively earn its keep, then it is our responsibility to keep the language in the public eye. If we don't do it ourselves then no-one else will do it for us; do you really want to spend your remaining days writing RPG-II?

Carrots

(Offered not to donkeys, but as some form of incentive.)

Every year the B.A.A. is going to look at the coverage of APL in non-APL activities and publications; when we find things that are really excellent in terms of advancing the cause, we're going to acknowledge the originators.

So, not only can you look forward to fame and fortune (sorry, we don't give those away), we might also give you a trophy. We're negotiating even now (more news next time).

But (and it's a serious proviso), we're not just handing out prizes to anybody – if all your papers and articles are dogs then we reserve the option not to award any prizes (this is the B.A.A., not Cruft's).

And the usual disclaimer of course applies – even if they could read and write, the B.A.A. Committee members are disqualified from getting a prize either while in office or for work done during their period of office.

(Editor: Presumably these are very special carrots – they help other people to see in the dark!)

The Programmer as Designer, The Designer as Detective

by Robert Pullman

Rule of Thumb 1:

> The more time one puts in design, the less time one puts in programming.

The assumption being that the design work progresses towards simplifying the problems that the system poses.

The essential problem that a system poses is how to satisfy all the constraints of the system simultaneously.

The first step, then, is to consider the constraints. Even on the smallest project, though, both the programmer and the client ought to share each.

Rule of thumb 5:

> The responsibility should begin with the programmer being in on the specification.

I think we're in 1987 now, a safe distance from the notion that programmers are oddball introverts, more comfortable with machinery than with people. We have all the natural urges and characteristics of the most "moyen" John and Jane. There is simply no reason why the programmer should not be there from the very beginning – taking notes, asking and answering questions, making suggestions.

The less experienced the programmer, the more important it is, at least to the programmer's long-term development, to be involved full-cycle in the project. Sure, it might be an array of confusion at first, but with time come poise and polish.

One of the worst predicaments to be in is to have responsibility without authority: authority as in author, as in author of the spec.

Conversely, if the programmer participates in formulating (formula: little beauty) the spec, and the brainstorming that precedes and surrounds it, the programmer can identify with the project.

Rule of thumb 6:

> There is always a tolerance range in a computer system. The client "lives with" certain shortcomings, enjoys certain surplus features. It never hurts to deliver more than was required. Sometimes a pittance of extra time on something seemingly uncalled for makes all the difference.

I've yet to write a system that implemented squeaky cleanly. I don't think that perfection in a system is any more feasible than the perfect automobile or cheesecake.

There are, however, essentials in the importance of the system. From a pure point of view they are mechanical in nature, something to be put in code and executed by a machine. From a practical point of view, the essentials reside in what the client could tell you, in what you know from experience to be required.

The key, as best as I can summarize, is not in some sort of combinational strategy for optimization. The key is in isolating the truly important requirements in their present sense, and satisfying them in a way that leaves ample room for meeting future requirements.

In short. . .
Rule of thumb 7:

> If you don't understand the spec, don't write the program. If you want (or have) to write the program, rewrite the spec so that you understand it.

And, nearly a corollary:
Rule of thumb 8:

> If you understand the spec, you have already solved the crucial design problems. The hardest design problems might be ahead of you, but you are on the right track.

APL SOFTWARE

A selection of currently available software:

* INTERPROCESS SYSTEM SOFTWARE — includes many significant enhancements to both VSAPL and APL2 including AP124 for APL2 and the very efficient 'AFM' shared file sub-system.
* PANEL Full Screen Manager
* PFS Program File System
* RDS Relational Database System
* POWERTOOLS/PC

Full training and support available with all our software.

For a more comprehensive list or details contact Philip Goacher on 03727 21282 at 27 Downs Way, Epsom, Surrey KT18 5LU.

The Road Not Travelled

by Sylvia Camacho

If a working lifetime is 40 years, I have already spent three-quarters of mine connected with the computer industry, being employed by it, married into it and, over the last 5 years, fighting for living space among proliferating micro-computers. This gives me more hindsight than most and, with the wisdom this is supposed to confer, I intend here to draw some conclusions.

Despite their reputation, there has been nothing spectacularly fast about the development and exploitation of computers. Mary Goldring pointed out that technological innovations, the motor car for instance, take about a generation to get from the drawing board to the mass market. Computer technology has been, in general, no exception although I think a case might be made that the increase in power per pound has been greater than for any other technology and will lead over the next 20 years to more extensive exploitation of micro-computers as components of other products than as general purpose calculating machines.

When, in 1965, I left computer manufacturing to start a family, the pattern for the use of computers as general purpose calculating machines was already set although there was still a debate in the trade over the rival merits of several (now called 3rd generation) programming languages. I left as magnetic disks were coming into use as a storage medium and although now larger and cheaper, they are still the preferred method of holding data. The advantage of disk over cards and tape is that portions can be overwritten leaving the rest of the data undisturbed. During the late 1960s and early 1970s video terminal equipment became cheap and reliable and this, together with the indexed files and databases made possible by disk gave rise to interactive program development and use. Changes since then in commercial computing have been largely consolidations and serial processor hardware performance is now close to physical limits. An increase in speed will require a change to parallel processing and, although the first experiments such as the transputer are in the laboratories, on the basis of previous experience we can expect this to take another 20 years to bring into widespread use.

I expect that my reader will by now be protesting that whereas the received wisdom is that we are at the beginning of a computing era, I am saying that we are near the end of one. This is difficult to reconcile with the recently buoyant market in micro computers and the attention of Government and Media. My reasons are that the scientists and engineers who set such a brisk pace in hardware innovation have not been equalled in imagination by software theorists. The attempt to use languages developed for linguistic analysis in the hope of realising something approaching 'artificial intelligence' is notably misplaced as it takes no account of what we know about the nature of the 'real intelligence' manifest in the brain. This, of course, does not stop vendors who need to bring a useful product to market advertising as 'artificially intelligent' software packages that are more properly called 'expert systems'. I believe that for the foreseeable future the bread and butter of computer system vendors will be packages, including expert systems, directed at a restricted market, developed by or in close collaboration with experts in the problems and practices of workers in that market. There is plenty of scope here for small enterprises but packages need a large market or a rich one and programming estimates being notorious for understatement, software development is likely to be viewed suspiciously by merchant bankers.

However successful the package market, computer users will need bespoke systems to handle those parts of their enterprise which are not 'run of the mill' and which give them a competitive edge. It is here that there is a major obstacle. The measure of it is the proliferation of data management packages widely called 4th generation languages. Turning over the pages of one or two trade journals will provide the names of 30-40 of these products, mostly directed at mainframe users and priced in tens or hundreds of thousands of pounds. As the market cannot support this number an unenviable job for computer department managers is to try to spot the vendors who will still be around in 10 years time.

In looking for reasons for this fragmentation, the history of software development is instructive. The transition from machine code (strings of ones and zeros) to a symbolic equivalent which the computer itself could translate into binary notation took only a few months to get started and a year or two to perfect. The need was great and although there was a difference of opinion about the choice of mnemonics for the various functions there were no issues of principle to be settled. What is more, as each manufacturer's machine had different registers, instruction sets & word formats, they each had to develop their own assemblers (2nd generation languages). The race to come to market with a language which would be a lingua franca between different machines led to much more acrimony but, perhaps because they were more aware of the importance of catching the market or perhaps because they were backed by the US defence establishment, the COBOL team produced the product which has been dominant for 20 years. For the sequential design of computer now used it cannot be challenged – the investment in software and programmer training makes wholesale conversion to any other language unthinkable. Theoretical arguments about language design are simply beside the point. I note that ADA, a language promoted by the Pentagon for real time applications and backed by our government with money for retraining, is finding an unenthusiastic response among software suppliers. They would rather use CORAL, the special purpose language they invented in the 60s, in which there is a small but stable pool of expertise.

If COBOL is so well placed, why are so many looking for an alternative? Programs whether or not written in COBOL will run faster as the speed of the hardware increases. One reason is that cheaper hardware has carried computers down-market where organisations do not expect to spend more on the programmers, whom they perceive as clerks operating the machines, than they do on the machines themselves. Unfortunately the speed of thought not having changed since COBOL was introduced in the 60s, a page of code takes about as long to produce as ever it did. What is more, the expectations of the customers has been raised by the use of package software and not least by recent, highly interactive micro systems produced for the mass market.

The heavy promotion of the many 4th generation products tempts data processing managers to use them in place of COBOL when new code is required. Many considerations influence them in this direction. They are told that the programs will require less code, that good work can be produced with less training, that the analyst himself can do the coding and stop acting as the middle man between the customer and the programmer and so avoid the delays and misundertandings that result from indirect communication. All of these claims are true but there are prices to be paid.

The relationship between 3rd generation languages and the machine code into which they are compiled is direct and easy to understand. A COBOL instruction to MULTIPLY is converted into a loop of many machine cycles to perform repetitive addition. Even extensions to the language used to access databases are easy to understand in principle. The instruction follows a succession of pointers 'navigating' among the inter-connected data-groups until the key of its 'search argument' is matched. Although most of the tedious detail of the machine code is hidden, 3rd generation code 'maps' quite closely to the code which it generates. This gives the coder a feeling for what he is telling the machine to do and the relative costs of different approaches. Moreover, he can choose between different paths to the same goal and plan the interactions between major sections of his job to minimise the time and resources used.

It is quite otherwise with 4th generation database managers, report generators, query languages and the like. The code provided is intended to specify what is to be done, not how the computer is to do it. Some people hope that this idea can be developed to the point where the customer will be able to dispense with the analyst as well as the programmer and generate his own code. It is seen that this would require a jargon-free language directed to the expression of the job to be done and making minimum concessions to the fact that it is to be run on a computer. The instruction input however still has to be translated into machine code only now without a programmer to interpret the 'what' into 'how'. Not surprisingly this lack of a directing intelligence often produces tortuous machine code. The difficulty of producing generalised algorithms to translate from loosely structured user code to robust and consistent machine code has led the vendors to concentrate on those facililities which are most in demand – data capture, report generation and file handling and enquiry. When used for these purposes and with a sustained attempt to keep the code simple, these tools are effective. They can be used to create some forms of bespoke system but they can never have the control over all the functions of the computer that is implicit in 3rd generation languages.

In 1957, six months out of University, I answered a small, classified advertisement in the Daily Telegraph: 'Wanted, keen and enquiring minds, knowledge of Boolean algebra an advantage'. I had read Philosophy because it excited me and Boolean algebra as part of the course in Formal Logic. It was my good fortune that my degree turned out to be vocational training! Thirty years ago I felt priveleged to be standing at a new frontier. What is the prospect now?

We seem to have reached an impasse. We want to produce software that is closely matched to the unique requirements of particular users yet the 3rd generation languages that have the power to produce such individually tailored software are expensive and so slow to specify that the requirement may have changed before the program is completed and attempts to modify it just start the whole slow cycle over again. On the other hand ingenious attempts to use 4th generation code to create sophisticated software are heavy on machine resource and make code hard to amend.

I think there are alternative roads which we might take but which of them we do take will depend on the view that influential members of the computing industry and the organisations that support them, both public and private, have of their enterprise. They can choose to see it as part science, part engineering which will progress only if effort is put into software research to match the effort that went into hardware. Alternatively they can choose to see it as a new craft which has now matured and requires only regulation and consolidation to turn it into a profession closely related to accountancy. I think that the former is desirable but that the latter is more likely and that users who are still looking for a tool to help them think may settle for a tool to help them keep their books.

The reason that I think computing is at the moment more craft than science will sound to a layman, paradoxical. It is that mathematics has been excluded from computing. In 1957 the Chief Programmer of my organisation was a Doctor of Mathematics. That was thought to be necessary to master the complexities of binary arithmetic. I was employed though I was not a mathematician (nor any sort of scientist) because I had to talk to customers and the Company 'knew' that scientists would be no good at that sort of thing. Mathematicians, of course, soon tired of hack coding and commercial computer departments were taken over by people literate but mostly innumerate. They are now the driving force behind the money going into 'artificial intelligence' research because they need software which will accept natural language instructions (preferably spoken, not typed). It is sad they will not live to enjoy it.

Mathematics is the queen of sciences because it makes possible the communication of ideas not easy to express concisely and unambiguously in extended prose. Advances in science are often accompanied by extensions to mathematics, particularly extensions to notation although these may not be invented by the man who breaks the new ground – Kepler managed without a multiplication sign! The inventors and beneficiaries of the new notation are often the inheritors of the new science and they use it to create a sound theoretical base for the next advance. The notation which makes the breakthrough and converts a craft into a science is not always mathematical. The modern science of chemistry owes much to John Dalton who drew up a table of atomic weights based on the proportions in which elements combined and used some crude diagrams to explain his hypotheses. As this approach seemed fruitful, others suggested a more systematic notation which has developed into the modern symbolism. Modern chemistry would not be possible without the formulae which underpin its concepts and are now so well known that H2O is a slang term and even H2SO4 is recognized by connoiseurs of limericks. Is there any sign of such progress in computing?

Some progress has been made in developing a systematic description of the raw material processed by computers. Data analysts have invented terms to describe abstract characteristics of data items such as the relationship between them. They set down the data structures to be processed as 'entity diagrams' showing items grouped and the one-to-many or many-to-many relationships between the sets. While drawing up these diagrams often helps define the requirements of the system, they cannot be manipulated like formulae. What is required is an analytic notation and preferably one which can be directly executed by a computer. As so much conventional commercial computing is concerned with comparisons between sets of data, the notation needs to express both mathematical and logical algorithms but I shall use 'mathematical' as shorthand for both.

So long as computer departments limit their ambitions to helping the accountants keep the bôoks a mathematical notation is unlikely to be adopted. Systematisation in accounting is confined to double-entry balancing and little mathematics is needed beyond powers for compounding and discounting and ratios for bringing performance in different departments or companies to a common base. However, accountants directly concerned in the management of companies need tools to allow them to project performance. This calls for statistical analysis and forms the basis of what is often called a 'Company Model'. There are, of course, other fields where modelling is appropriate from Astronomy to Zoology and such models are usually coded in a 3rd generation language like FORTRAN or more recent 4th generation statistical packages like SAS. Unfortunately such code, although executable, is not analytic. Using the analogy with chemistry these are the languages of alchemists. They also have all the drawbacks identified above. Therefore accountants needing models are abandoning the mainframes and learning to adapt micro-computer packages, usually some variety of spreadsheet. This has the advantage of being 1 or 2 precent of the price of mainframe software and allows them to by-pass the rigidities of DP departments. However, to get the power necessary to model a large company may require a network of micro-computers and file-servers and needs technical know-how in the complexities of micro-computer hardware.

My wisdom of hindsight is therefore that users will continue to be frustrated until they use a mathematical notation to analyse and specify their requirement and a notation that is executable to enable the computer itself to prove their algorithms. Micro computers currently share the language limitations of the mainframes and have additional limitations in speed. However, the new, cheap mass market will respond more quickly than the traditional, expensive and small one. This should enable an analytic and executable language to penetrate fast and deep. Is there any reader who thinks I am saying there is no road worth travelling until someone has invented such a notation for computer science? I am not saying that because there is a notation. Thirty years ago it was called 'Iverson's Notation' but now it is known as APL. To penetrate it must of course, like chemical notation, be taught in every school. Unfortunately, while textbooks with non-alphabetic notation are more expensive than others, they are not prohibitively so but as yet there is no cheap, readily accessible APL interpreter for the most common micro-computers. However if my ambitions are realized we shall have one long before I end my working career.

TECHNICAL SECTION

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

Technical Editorial: Mind your language

by Jonathan Barman

What other languages should APLers learn, and why? There is concern that the APL community is closed and does not take account of the 'real' computing that is going on. APL is, after all, a tiny part of the overall computing activity. Naturally, as this is an APL periodical, we think that it should play a much more important role.

Previous editorials have raised the issue of where APL should or should not be used, and we were hoping that articles would be written on the subject. So far there has been very little, and we feel that one of the reasons could be that there are very few APLers who are comfortable in using another language.

If this is true, then it could explain why APL is such a small part of the computing world. If someone does not know another language reasonably well, and is not aware of the way in which the majority of computing is carried out, it is then very difficult for them to describe the benefits of APL in a rational and convincing manner.

Standard APL is a non-looping language. Part of its charm to those who love it is the amazing solutions that are available where, at first sight, looping would appear to be essential. The 'for each' operator in the extended APLs allows looping solutions to be defined naturally, but extended APLs need lots of memory and computing power. If a looping solution is the easiest to implement, and extended APL is not available, what are the alternatives? BASIC is so horrible after the elegance of APL that most of us shudder at the thought, although compiled BASIC is quite fast. In fact pretty well all the options come down to a compiled language. PASCAL and C seem to be the most popular alternatives. C seems to have the edge in our view as you can, if needed, get down to basic machine operations, and it is easy to prepare lots of modules which can be pulled off the shelf when needed. This is similar to the the way in which functions can be copied from a utility workspace in APL. PASCAL is used for teaching programming and may be the best for a relative novice. Many APLs now allow ASSEMBLER routines to be called, and if the added speed is worth the considerable investment in programming time, then it is relatively easy to include special functions in the workspace.

There are masses of specialised languages with much the same level of use as APL, some of them such as PROLOG are much further away from the normal computing traditions. Which ones should we be looking at, and what are the problems that they can solve more easily than APL?

Fourth generation languages are much in the news, and we need articles about them so that we can see if they really do offer the advantages claimed, and if we should seriously consider their use in place of APL for specific problems. Has any reader of VECTOR ever used a fourth generation language?

Please write to give us your personal experiences of successful (and unsuccessful) use of other languages, particularly their use alongside APL, or in applications that might be considered 'natural' for APL.

Technical Correspondence

APL2 bug update

From David Piper

18 February 1987

Sir,

Following my letter to the last edition of VECTOR, I am pleased to say that IBM have seen fit to cure some of the bugs I reported in APL2. Three of the four bugs I reported last time have been fixed. The fourth is claimed not to be a bug anyway, and who am I to argue.

The following problems have been fixed:

- Fractional line numbers in Editor 2.
- Problems with APL WSID after)COPY.
- Problems with APL QUIET (IBM claimed this was not a bug, but then fixed it).

I have also found some undocumented changes in the most recent release of APL2 (Version 1 Release 2). Since these may be of interest to other users, a brief resume follows.

1. QSAM AP 111

The QSAM AP has been upgraded to allow multiple writes to be driven from one assignment to the RECORD variable. The cover functions I have already implemented allow advantage to be taken of this feature simply by enclosing the entire left argument (see my article in the VECTOR Vol.3,No.3).

If a text array is assigned to the record variable, each line of the array will be processed as a record, WITHIN THE AP. I have done some benchmarks, and it looks as though this enhancement can save upto 75% of the CPU associated with writing a file. Unfortunately this technique cannot be easily used with the VAR conversion option, variable length records. No enhancement has been made to reading data.

2.)CHECK system command

This system command allows various invocation options of APL2 to be changed 'in-flight'.)CHECK SYSTEM displays the current settings.)CHECK SYSTEM option (value) will change the value associated with the relevant option.

3.)CHECK TRACE

If the SYSDEBUG level is set above 128 (using)CHECK SYSTEM), a new type of trace can be placed on functions. This trace counts the number of times each line of code is executed and the total cpu time used. The trace can be set up as follows:

```
)CHECK SYSTEM SYSDEBUG(128)
)CHECK TRACE TIME
17 ⎕IB 'FOO'        ⍝ Set the trace option on function FOO
19 ⎕IB 'FOO'        ⍝ Check the initial values of the counters
```

The function (FOO in this case) can now be used normally. Any further reference to 19⎕IB 'FOO' will display the updated trace results. Note that if either of the)CHECK commands is omitted, the trace will not work. In fact trying to reference ⎕IB without setting SYSDEBUG > 128 results in a SYNTAX ERROR.

I hope the above will be of interest to other users of APL2,

Yours sincerely
D.B. Piper
Cocking and Drury, London.

Use of ⎕WIN

From Mr A N Wiggins

Sir,

I have just read Martyn Adams review of APL*PLUS release 6. I agree with his comments about the ⎕WIN functions and their descriptions in the manuals.

Maybe the following function will be of help to any readers trying to get to grips with this area.

```
    ∇ RES←SCN ARG;INP;WND
  ⍝∇SCN - CREATE OR EDIT A FULL SCREEN PANEL : A N WIGGINS 12 FEB 87
  ⍝ THE GRAPHICS CHARACTERS ARE IN THE 'F' KEYS
  ⍝ WHEN YOU ARE STAISFIED WITH THE SCREEN, EXIT USING THE <INS> KEY
  ⍝ THIS IS WRITTEN AS A BASIC DESCRIPTION OF THE WAY IN WHICH
  ⍝     THE ⎕W FUNCTIONS WORK ... AND NO MORE !!
  ⍝ E.G.  RES←SCN ''  ⍝  START CREATING A NEW SCREEN
  ⍝ E.G.  RES←SCN RES ⍝  EDIT AN EXISTING SCREEN

   GKEYS ⍝                 AVAILABLE IN THE SUPPLIED <APLDEMO> WORKSPACE
   ⎕TCFF ⍝                 CLEARS THE SCREEN
   WND← 0 0 25 80 ⍝        NEW WINDOW SHAPE
   WND ⎕WPUT ARG ⍝         PUT UP THE SCREEN
   0 0 ⍴¯1 ⎕WKEY 433 ⍝     REDEFINE THE <INS> KEY (SEE BELOW)
  LOOP:INP←⎕WIN WND
   →(433≠INP[2])/LOOP ⍝    CAN ONLY BREAK OUT OF LOOP WHEN <INS> IS PRESSED
   RES←WND ⎕WGET 1 ⍝       COLLECT THE WINDOW AND OUTPUT
    ∇
```

Yours,

A N Wiggins,

Lombard North Central,
Lombard House,
London, W1A 1EU.

APL Trivia

Wimbock-APL Application Notes

by 'Dan Wimbock'

Some of the technical features of this little-known implementation were described in an earlier edition of VECTOR (April 1985); regular readers will be pleased to know that Dan has not been idle over the past two years – indeed he was unable to exhibit at APL86 due to pressure of other commitments.

The major thrust of development has been not so much in advancing technological features of the interpreter (although the 'airborne' generalised array implementation is an advance on both 'floating' and 'grounded' which is worth an article in its own right – maybe next year); rather it has been in emphasising useability and the pragmatic nature of this particular incarnation of APL.

An instance of this combines Dan's interest in both Expert Systems and a new approach to 'help' facilities within applications.

When we look at the conventional 'help panel' approach we find a number of problems which limit the possibility of actually providing the assistance which our user really wants. . .

a) Information displayed is in a quite rigid format – once seen it can only be re-seen; if it fails to create understanding there is rarely an alternative phrasing which could shed light.

b) Analysis of the user's problem is the user's responsibility – the system doesn't attempt to gain insight into why help is needed.

Dan's view of the solution to this problem also falls into two parts,

a) The ability to construct meaningful and grammatically-correct sentences in the user's own language.

b) A diagnostic facility which can analyze underlying problems.

Taking the first, consider a sentence encountered very early in the education process of most English-speaking humans – yet one which is only uttered under the rarest of circumstances by the most advanced computing hardware. . .

'The cat sat on the mat'

Grammatical structure is subject-verb-object; a structure which is able to tolerate considerable distortion in terms of additional qualifiers (adjectives and adverbs), plurality ('the cats sat on the mats') and abstraction ('a cat sat on a mat'). This single sentence structure is capable of embracing an immense diversity of subject matter (cats, dogs, mats, rugs, even computers) and the most advanced of technological concepts. . .

'The X-195A computer architecture is based on 73-bit Gallium Arsenide VLSI technology'

Dan has therefore implemented a help-message-generation-facility (HMGF) which is based on this expert-system-paradigm (ESP) using Appropriate Vocabulary (AV), with pseudo-random phrase selection (PRPS) as a methodology for perceptual diagonalisation of the underlying message (invocation of the facility successively under the same stimulus does not generate repetition of the message, nor does it take the user to a deeper level before understanding of a first-order message is attained).

Taking as an example the sub-phrase structure illustrated below. . .

```
      DPY GOBLIN
.→3------------------------------------------.
| .→2-------------. .→2-------------. .→1-----. |
| | .→3-. .→6----. | | .→3-. .→5---. | | .→3-. | |
| | |THE| |A VERY| | | |OLD| |SMALL| | | |ELF| | |
| | '---' '------' | | '---' '-----' | | '---' | |
| 'ϵ---------------' 'ϵ---------------' 'ϵ------' |
'ϵ-------------------------------------------'
```

This example uses the APL2-emulation mode of Wimbock-APL:

```
⎕APL←'APL2'
```

DPY is from Brown (1986)

The PRPS algorithm is simplicity itself:

```
     ∇Z←PRPS TEXT;⎕RL
[1]  ⎕RL←×/¯3↑⎕TS~0
[2]  Z←▼(?ρ¨TEXT)⊃¨TEXT
     ∇
```

Hence:

```
      PRPS GOBLIN
THE SMALL ELF
      PRPS GOBLIN
A VERY OLD ELF
```

The system designer needs only feed in a suitable vocabulary in order to create a full and varied selection of relevant help messages; future extensions planned at Wimbock Enterprises include automatic extraction of vocabulary from system design documentation and utilisation of the ⎕NLT system variable to create national-language customisation (although this latter may require some work on the Generalised Grammar Interface – GGI).

The other half of the solution entails straightforward analysis of the)SI stack to determine the context of the user's problem, and this is combined in the full implementation with a pseudo-ELIZA dialoguing technique to elicit the user's state of mind. This is presently the subject of a patent application and Wimbock Enterprises are unable to divulge full details of the techniques and algorithms used; however Dan feels that the averagely diligent VECTOR reader should be able to devise a suitable solution fairly quickly (it is unlikely that using the full capabilities of Wimbock-APL the necessary function would be longer than <PRPS> above).

Reference:
*Algorithms for AI in APL2;*Brown, APL86 Tutorials
(Editor: Gettem while we gottem).

Recent Enhancements to Wimbock-APL:

Dan has just announced a new release, containing a number of enhancements among them being . . .

⎕APL System Function

Ensures application portability from other environments; ⎕APL may be set to values 'APL2', 'IPSA', 'STSC', 'VSAPL', 'APL68000', 'DYALOG' in addition to default setting 'WIMBOCK'. When set all symbols are interpreted in the context of the parent implementation. Attempts to set ⎕APL to an unrecognized value will result in a DOMAIN ERROR, e.g.

```
              ⎕APL←'IAPL'
      DOMAIN ERROR
              ⎕APL←'IAPL'
                 ^
```

Monadic iota domain extended

Negative integers:

```
                     ⍳¯5
      ¯1 ¯2 ¯3 ¯4 ¯5
```

Complex integers:

```
             ⍳2J2
      1J1  1J2
      2J1  2J2
```

Further extension to non-integer arguments are anticipated 'soon'.

'And So Forth' Operator

This is a new niladic operator intended to improve productivity at the function definition stage; it is only available on the dedicated Wimbock-APL Machine and takes advantage of the advanced AI features of the machine. At any point during function definition when it is obvious to the programmer how to continue the 'and so forth' operator may be invoked, for example:

```
[12] ABC←1 2 1⍉XYZ
[13] XYZ←⌽¨⍳0J10
[14] ...
[21] ⎕←⍴XYZ
```

Control of function definition is regained by the programmer at the explicit line number entered. This feature has been incorporated to make Wimbock-APL more attractive to conventional systems analysts and Dan wishes to express his gratitude to his supervisors in earlier years for making him aware of the usefulness of this notation.

'Circle' reclassified as an operator

This is an enabling move for a further generalisation of array structure (away from rectangularity to a more general ovoid form); in the present implementation its usefulness is limited, but for example

```
      90⌽○2 2⍴⍳4
3 4
1 2
```

Note that this assumes prior use of the ⎕ANGLES system variable which allows user choice between radians and degrees as measure of rotation.

Note also that domain is severely restricted with present array limitations

```
      87⌽○2 2⍴⍳4
DOMAIN ERROR
      87⌽○2 2⍴⍳4
         ^
```

Prize Competition: Sweeten your Combinations

by Derek Wilson

A popular brand of sweets comes in coats of many colours, and is sold in tubes and packs. These are filled at random from the set colours. What is the probability that a tube will be missing one or more colours?

The problem is to write a function which takes a left argument of the number of sweets in a pack and a right argument of the number of colours, and returns a vector of all the probabilities. The first element is the probability that there will be exactly 1 colour in the pack, the next element the probability that there will be exactly 2 colours, and so on. For example, if there were only 5 sweets in a pack and 3 colours available, then the result would be as follows:

```
      5 COLOURS 3
0.01234567901 0.3703703704 0.6172839506
```

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 any functions were written.
- The date and the entrants' full name and address must appear on each sheet of the 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.
- Members of the B.A.A. committee, activities working group or journal working group are ineligible.
- DOS format diskettes containing APL*PLUS, IBM or SHARP APL workspaces are acceptable; diskettes will be returned.
- Unless otherwise stated, entrants should submit only one entry. We encourage submission of alternative approaches, but the entrant must indicate clearly which one answer is the entry in the competition.
- Non-members of the B.A.A. are encouraged to enter the competition. If they should win, then part of their prize will comprise free B.A.A. membership for the current year.
- Late competition entries may be accepted if the competition has not yet been judged.

Introduction to Technical Contributions

Our first contribution comes from the statistics department of the University of New South Wales, Australia. J B Douglas responds to Dick Bowman's domino difficulties (See VECTOR Vol.3, No.1) with a short piece entitled 'Polynomial Curve Fitting'.

In the same issue of VECTOR, John Sullivan wrote about "Fast Fibbing". His article has stimulated two replies from our readers. Joseph de Kerf – the Chairman of the Belgian APL-CAM Users Society (BACUS) has contributed his thoughts on the subject. Alan Sykes has written from the Department of Management Science and Statistics at the University College of Swansea, providing an APL2 view of the subject.

David Piper is a consultant with Cocking and Drury; his latest foray into the depths of APL2 is entitled 'Using Name Association for Data Translation'.

Paul Chapman is an independent consultant who is currently employed by the I-APL team to write the free APL interpreter for schools. Paul has made his arduous and lonely task more bearable by writing an I-APL diary, the first instalment of which is published in this issue of VECTOR. Watch out for more diary instalments in future issues!

Call for Technical Articles

If you are interested in publishing an article in the technical section of VECTOR, please contact David Ziemann or Jonathan Barman, the technical editors. Our backlog of articles has now been exhausted, so we can guarantee to get you in print immediately.

If you receive VECTOR regularly, you will appreciate the articles and correspondence created by others. You are of course familiar with your own work and views, but others are not; please consider sharing your knowledge with the other readers by making a contribution of your own.

Please note also that we are trying to orient future issues of VECTOR around more specific themes. Obviously we are particularly keen to receive articles based on these topics. Our current plans for volume 4 of VECTOR are as follows:

Number	Copy date	Issue date	Theme
1	24 April	July 87	Graphics and user interface
2	24 July	October 87	Applications
3	16 October	January 88	Education

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

Polynomial Curve Fitting

by J B Douglas

I refer to Dick Bowman's and the Editor's comments *(VECTOR Vol.3, No.1, pp 99-100)* on the different responses of different APL systems to a dyadic use of domino for polynomial curve fitting.

Dick's remark, that it is 'interesting' when the same code applied to the same data gives different results, is certainly valid. But without downplaying the importance of having good algorithmic and numerical procedures ("the code"), I should like to emphasize the other aspect, the data, of what is a perpetual problem. The "perpetual problem" is to devise methods sufficiently robust for appropriate analysis of any data; it is perpetual because whatever method may be devised it is always possible to invent data for which the method will misbehave.

What is often sought is a ". . . fitted curve . . . to look nice and smooth . . ." So let's graph Dick's data (see Figure 1). The curious behaviour at odd and even "years" is obvious, and no meaningful smooth curve seems plausible – in particular, a cubic cannot possibly reproduce the marked oscillation of the Y-values at the later "years" X. To obviate readily removable DOMAIN or LIMIT ERROR difficulties, which may arise simply because large numbers like 1995 are raised to high powers in the calculations, subtract 1984 from X. This gives numerically reliable regression coefficients and a smooth fitted curve (see Figure 2) which is basically nonsense when thought of as representing the data. (APL*PLUS PC is capable of giving an equivalent to this result without having to subtract the 1984.)

A more detailed analysis shows that of the four regression coefficients, only the first, associated with the constant term, is "large" (compared with its standard error) and likely to be meaningful, and that the mean square about the fitted cubic really isn't much less than that about a constant fitted value.

The numerical results are as follows, with the data first:

```
X:  1  2  3  4  5  6  7  8  9 10 11
Y: 37 36 39 38 41 39 43 38 43 35 42
```

```
+B←Y⌹A←X∘.*0 1 2 3
```

returns the regression coefficients

```
          34.212   2.040    -0.225   0.00777
```

for which the standard errors are:

```
          5.1      3.5       0.67    0.037;
```

fitted values are calculated from `A+.×B`

Inspecting the data, the graph suggests that two distinct quadratics would represent them well at the years recorded: one quadratic for the odd, and another for the even, years. This fitting can be achieved by replacing the matrix A by the matrix C; both are shown below.

$$
\begin{array}{cccc}
\multicolumn{4}{c}{A} \\
1 & 1 & 1 & 1 \\
1 & 2 & 4 & 8 \\
1 & 3 & 9 & 27 \\
1 & 4 & 16 & 64 \\
\ldots & & & \\
1 & 11 & 121 & 1331
\end{array}
\qquad
\begin{array}{ccccc}
\multicolumn{5}{c}{C} \\
1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 2 & 4 \\
1 & 3 & 9 & 0 & 0 \\
1 & 0 & 0 & 4 & 16 \\
\ldots & & & & \\
1 & 11 & 121 & 0 & 0
\end{array}
$$

Then executing:

```
        Y⌹C
```

returns five coefficients all of which are large compared with their standard errors, the fit being displayed in Figure 3. (The coefficients and their standard errors are:

34.141	2.082	-0.1234	1.632	-0.1507
0.72	0.29	0.024	0.31	0.029)

So in some sense this smoothly, and quite closely, reproduces the data. It leaves uncertain what meaning should be attached to interpolated values – but so do the data.

I believe the principal message to be that one should always look at the data and what any procedure predicts for the data, to see that the question and the answer are reasonable. The 'misbehaviour' of domino is minor compared with the mis-specification of a cubic for the representation of the data.

Figure 1. Bowman's original data

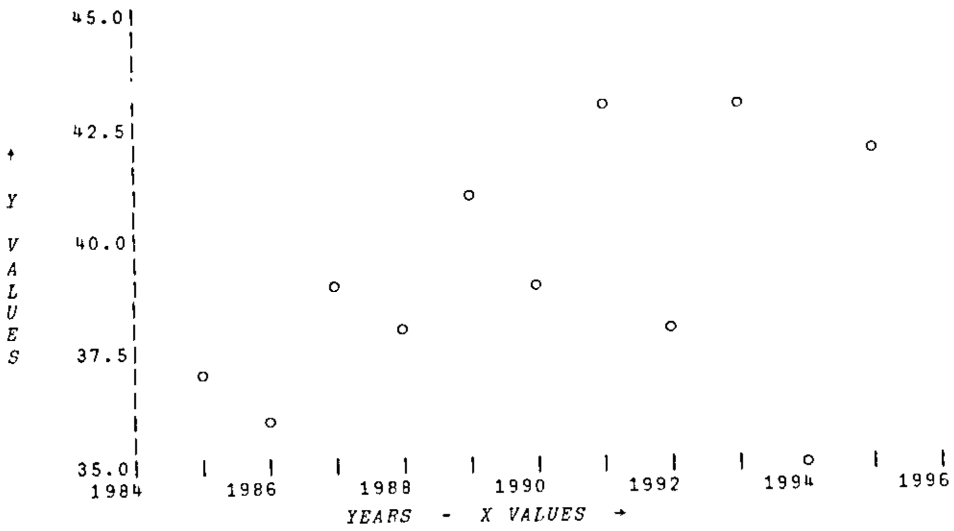


Figure 2. Original data with a fitted cubic

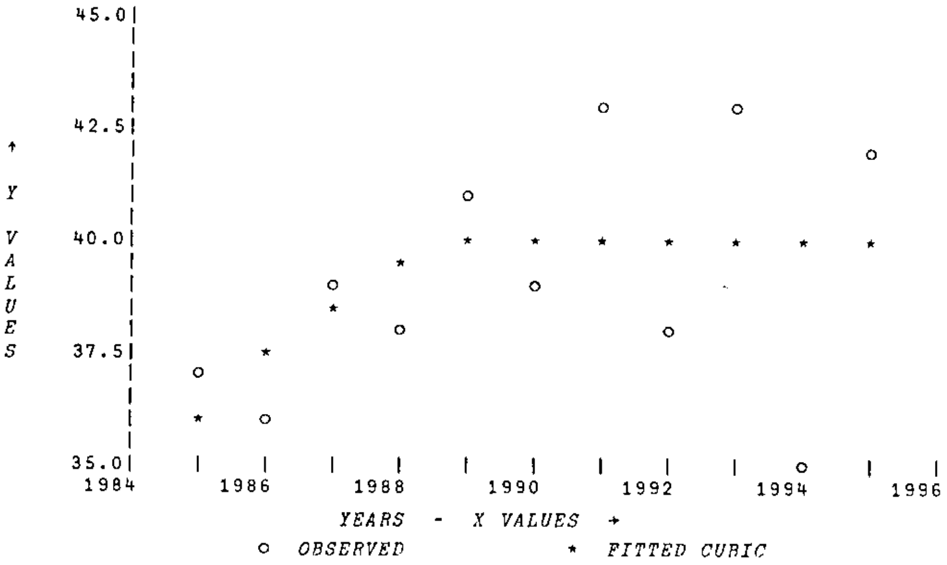
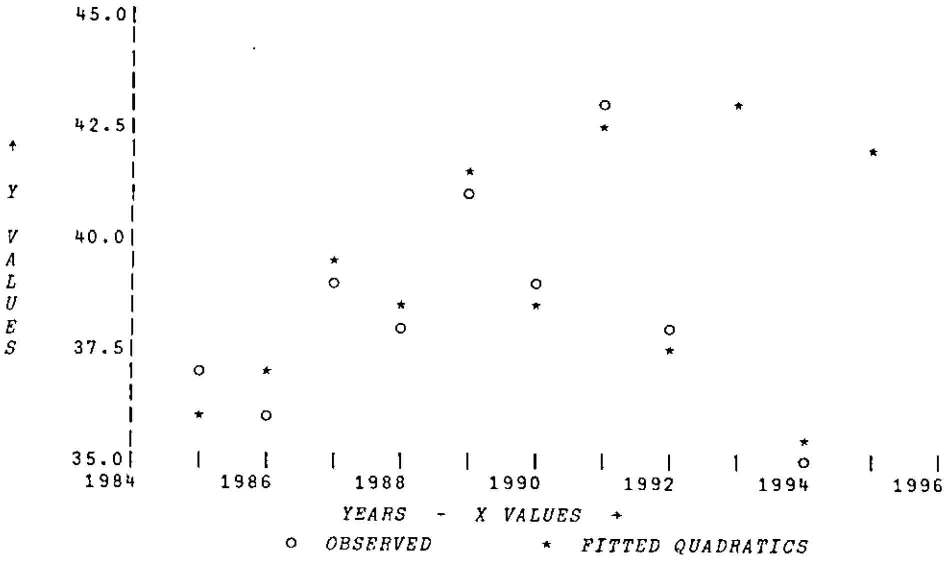


Figure 3. Original data with a fitted pair of quadratics



More on "Fast Fibbing"

by Joseph de Kerf

In a previous issue of VECTOR (see ref 1), John Sullivan compares the Fibonacci function FIB1, based on the recurrence relation:

```
   ∇ R←FIB1 N
[1] R←1 1
[2] →(N>⍴R←R,+/¯2↑R)/⎕LC
   ∇
```

with the loop-eliminating Fibonacci function FIB2:

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

The defined function FIB2 is based on the well-known golden section formula:

$$u_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$$

The second term of the formula is very small, even for lower values of n, and converges to zero. By setting:

$$u_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n$$

we get:

n =		u =		
	1		0.7236	
	2		1.1708	
	3		1.8944	
	4		3.0652	
	5		4.9597	
	6		8.0249	
	7		12.9846	
	8		21.0095	
	9		33.9941	
	10		55.0036	etc ...

Rounding off to integers gives the exact values for the Fibonacci Numbers. This means that we can replace FIB2 by the Fibonacci Function Mark 3:

```
   ∇ R←FIB3 N
[1] R←⌊0.5+(÷R)×(0.5×1+R←5*0.5)*⍳N
   ∇
```

Apart from some overhead, the execution of FIB3 versus FIB2 is speeded up by an extra factor of about 2.

This trick is well known in the literature (see e.g. ref 2). So, to the recommendation "Always do the systems analysis before you write the program", we should add "Consult the literature before you do the systems analysis."

References.

1. *Fast Fibbing*, by John Sullivan, *VECTOR* 3.1, July 86, pp 113-114

2. *Motivating Arrays in Teaching APL*, by Garth H Foster, *Proc. of 5th International APL Users' Conference, Toronto, May 1973*, pp 3-1/3-8. Theme: APL applications in Education and Business. Printed by Canadian Printco Ltd., Toronto, 1973.

More Fibbing

by Dr Alan Sykes

John Sullivan's article (VECTOR Vol.3, No.1), raises some important issues concerning APL and its relationship with other languages. APL provides only rather inefficient tools (some more inefficient than others!) for solving those computational problems typified by the Fibonacci sequence. The most natural method is to consider looping (or function recursion) and the search for alternative methods, whilst intellectually stimulating, often give computer scientists (justified?) cause for concern! After all, John Sullivan's solution (FIB2) isn't recursive, nor does it involve only integer arithmetic.

However, to avoid such criticisms completely, it would be necessary to write our program in machine-code – undoubtedly the 'best' language for recursive integer arithmetic! Many computer scientists would not readily follow this course, and clearly some compromise between the efficiency of the program execution and the efficiency of the program construction is usually made. What we APLers must do is to show our readiness to construct auxiliary processors for those applications which demand them, thereby showing our computer science colleagues that we can "have our cake and eat it".

In a wider context of computing solutions to difference (or differential) equations, John Sullivan has a point. Numerical solution of even simple difference equations can be fraught with the likelihood of numerical instability. Hence, if an exact solution is known, this may lead to an efficient and error-free method of computation. In this context, FIB2 is a good solution.

Incidentally, at the risk of invoking my own criticism, may I present an APL2 solution (which I hasten to add is much slower than FIB2). FIB3 calculates the first 2N Fibonacci numbers:

```
   ∇R←FIB3 N
[1] R←,(↑+.×\N⍴⊂2 2 ⍴ 1 1 1 2)[,,1]
   ∇
```

This example is interesting in the context of becoming familiar with useful 'second-generation' APL idioms. It is inefficient because the powers of the matrix

```
      1    1
      1    2
```

created by

```
+.×\N⍴⊂2 2⍴ 1 1 1 2
```

are not created recursively. The 'scan' operation '\' operates recursively with some binary operations; for example

```
+\⍳ 10000
```

is much faster than

```
-\⍳ 10000
```

because '+' is 'associative' whereas '-' is not. Although '+.x' is an associative operation on matrices, the APL2 interpreter does not recognise this and hence the inefficiency.

I am very grateful to John Scholes of Dyadic Systems for his helpful comments on FIB3 during APL'86 at Manchester.

APL CONSULTANTS

LONDON & READING

Account Managers	(6 years+)	to	**25K**
Senior Consultants	(4-6 years)	to	**21K**
Consultants	(2-4 years)	to	**17K**
Junior Consultants	(1-2 years)	to	**13K**

Are your APL skills and potential being recognised and rewarded?

Cocking & Drury consultants have been implementing successful decision support applications for 10 years, with clients who appreciate the productivity benefits of APL.

In our professional team you will experience a range of APL environments - APL*Plus, VSAPL, APL2 and Unix, on both mainframes and micros. You will also be developing systems which, increasingly, need to interface with non-APL Information Centre products.

Of course as the leading APL consultancy, in a rapidly expanding market, we offer a rewarding career with first class benefits - profit sharing, free health insurance, and a non-contributory pension.

For further details call Ralph Wilson on 0734 588835

COCKING & DRURY LTD.

155 Friar Street, Reading, RG1 1HE

Using Name Association for Data Translation

by David Piper

1. Introduction

When used in a commercial data processing environment, systems implemented in APL will often be required to address significant volumes of data from 'external' (i.e. non-APL) files. Since these files are not generated by APL code, they may well contain data in formats not readily accessible by APL. Prime examples of such formats include packed decimal and the IBM S/370 floating point format.

One of the supplied workspaces in both the VS APL and APL2 environments contains functions for translating external data into APL numeric form. Since these functions are coded in APL, making use of some of the most complex APL primitive functions (index-of and encode/decode), execution speeds tend to be somewhat slow. This poor performance is of especial concern when large volumes of data are being translated.

2. Name Association (⎕NA)

⎕NA is a system function provided as part of APL2 (Version 1 Release 2 and later). It is designed to allow routines coded in languages other than APL and stored outside the workspace to be invoked in exactly the same way as APL functions.

The functions are invoked by use of an associated processor, one for each language type being used. In the CMS environment, two associated processors are provided; 10 allows use of the REXX language, 11 allows languages compiled into machine code to be called. TSO provides only processor 11, since there is no equivalent to the REXX language.

3. Properties of Associated Objects

One of the key properties of associated objects is that they obey all the rules associated with 'normal' APL2 objects:

- They appear in ⎕NL,)NMS etc.
- ⎕NC returns the correct name class for the object.
- Associated functions and operators can generate APL2 errors.
- The association is preserved in a)SAVEd workspace.
- Associations can be)COPYed from another workspace.
- Associations are dissolved by)ERASE or ⎕EX.
- ⎕AT returns the correct information (even a timestamp can be defined.)

An associated function appears identical to a locked function as far as the application programmer is concerned.

4. Interface Definition

The interface between APL2 and the associated object is defined in a 'names' file. Amongst other factors, the names file contains descriptions of:

- The name of the associated function.
- The load library where the function can be found.
- The module name and entry point if different from the associated name.
- Interface type (object, fortran or function).
- Descriptions of the arguments, including structure and data types if these can be anticipated.
- A description of the function and time-stamp.

The most complex feature to be defined is the pattern of the arguments. The pattern definition must include details of structure – that is the pattern of nesting in the argument. Details of rank, shape and acceptable domain must also be included. To facilitate the implementation of assembler coded functions, translation options can also be specified. These will cause an argument to be translated from APL format to S/370 on input to the function and the result to be re-translated on output.

5. Supplied Functions

With APL2 several functions coded in assembler are provided. The lack of documentation makes these more difficult to use, several of the functions are, however, extremely useful.

Several of the functions allow a vector to be partitioned in various ways (see also my letter in VECTOR Vol.3, No.3). These include:

CAN – partition on zeroes in a boolean vector.
DAN – partition on delimiters, removing the delimiters.
SAN – partition on delimiters, retaining the delimiters.

Other functions are more esoteric such as SVI which allows a systems programmer to investigate which associated processors are functioning. The functions used for converting data between APL and S/370 formats and back again are RTA (record-to-array) and ATR (array-to-record).

6. RTA and ATR functions

These two functions translate S/370 data to APL formats and back again respectively. The left argument takes the form of a pattern specification of the data to be translated. The pattern specification is similar to that used to describe the left and right arguments in the associated functions name file definition.

The RTA and ATR functions are capable of translating an entire record, with fields of varying lengths and types at one call. The cover functions presented here are more limited, they are designed to replace the IBM functions in 1 UTILITY. Each function translates any number of occurrences of a given data type into APL format (or vice versa).

For example, when read from a file, a series of 6 byte packed fields will each be represented as 6 APL characters from ⎕AV. In this form, the data cannot be used by APL. Using the PDI (packed decimal input) function the 6 character representation is converted into valid APL numerics. Supplying an N by 6 array of packed decimal data will result in a numeric vector of length N.

7. Pattern Arguments

The key factor in performing the data translation is the creation of the pattern argument. The pattern argument is of two separate forms, depending on the direction of the translation.

When translating from S/370 into APL, the input data is always supplied to the RTA function as a vector. The rank and shape of the output is implied by the rank and shape of the argument to the cover function.

The first part of the pattern argument defines the translation option. The option is in two parts, the data type (e.g. 'P' for packed, 'I' for integer) and the number of bytes per field ('P8' will translate 8 byte packed decimal data). The next part specifies the rank (always 1) and length of the resulting APL vector. The resulting numeric vector is reshaped into an array that conforms with the input argument.

When translating from APL to S/370, the input data is supplied as an array, the result is always a character vector. In order to be the inverse of the 'input' cover function, this vector is reshaped. The pattern argument is created slightly differently. The first item is an asterisk (*) stating that the count of items to be translated is unknown. The next item describes the translation option (as above). The last item is another asterisk, informing that the rank of the array being translated is variable.

8. Efficiency

The main aim of using the assembler-coded functions is to gain performance and execution speed. The following benchmarks show just how effective the assembler-coded functions are relative to the APL equivalents. The tests are performed by averaging over 1000 conversions of a specified array.

For the BIT conversion functions, 256 bytes of data are converted each time. For the PACKED and INTEGER conversions, integer arrays of 24 rows by 10 columns are converted. Both negative and positive data are included. Floating point conversion uses a 24 by 10 array of floating point numbers, of both negative and positive sign.

Table 1

Converting from APL to S/370
(CPU milli-seconds per array conversion)

Type	⎕NA	APL	⎕NA/APL (%)	Rank
Logical (Bit)	2.68	20.13	13.31	2
Packed Decimal	3.48	23.28	14.95	3
Integer	2.58	11.39	22.65	4
Floating	2.56	210.58	1.22	1
Average	2.83	66.35	4.27	

Table 2

Converting from S/370 to APL
(CPU milli-seconds per array conversion)

Type	⎕NA	APL	⎕NA/APL (%)	Rank
Logical (Bit)	2.23	20.52	10.87	4
Packed Decimal	3.68	72.88	5.05	3
Integer	2.09	101.35	2.06	1
Floating	2.50	77.24	3.24	2
Average	2.74	68.22	4.02	

9. Function Changes

The functions listed below are as identical as possible to those supplied in 1 UTILITY. A few changes have been forced by restrictions in the ATR and RTA functions:

- The II and IO functions will now only accept half or full word integers.
- The FO function is capable of producing single precision as well as double precision floating point output.

The integer processing restrictions imposed are not very serious. Nearly all integer fields are either half- or full-word. Additionally, the performance of the II/IO functions in 1 UTILITY is so poor with integers greater than 4 bytes, that processing any volume of data is made impracticable.

The FI function in 1 UTILITY is capable of converting full or double precision floating point into APL numerics. The FO function only produces double precision output. Single precision can be produced by 'dropping the last four columns of the output' (quote from comments in the function). This technique is obviously inefficient. By default, the FO cover function will produce double precision output. The required precision can be specified in the left argument.

10. Conclusions

The requirement to translate data between S/370 and APL formats arises quite frequently in the commercial data processing environment. The use of assembler coded functions to speed up such translations makes the processing of significant volumes of data a realistic proposition.

The functions given are intended to cover the use of the IBM supplied functions in 1 UTILITY. To this end, their syntax and functionality is mirrored as closely as possible. The RTA and ATR functions could easily be utilised in a different manner which would further facilitate data conversion work.

The documentation of the supplied assembler coded functions is very poor. Improvements in this area would make use of the supplied functions much easier.

11. Sample Functions

```
[⍝]∇ FI.3  ⍴: 11  1987-02-19 13.04.32
[ 0]     R←FI DA;ND;⎕IO
[ 1]    ⍝ FN: Convert from S/370 floating point to APL2 numeric
[ 2]    ⍝ DA - Character array                           S/370 Floating point data
[ 3]    ⍝ R  - Numeric array                             APL2 equivalent data
[ 4]     ⎕IO←1
[ 5]     ⎕ES(~(≡DA)∊0 1)/5 4                           ⍝ Error not simple.
[ 6]     ⎕ES(~(ND←¯1↑1,⍴DA)∊4 8)/5 4                   ⍝ Full or double words only
[ 7]     →(3 11 ⎕AT 'RTA')/ok                          ⍝ Create/check association
[ 8]     ⎕ES 1 2                                       ⍝ Unable to associate
[ 9]    ok:R←'(E5 9 5555559)'⍕ND,1,×/¯1↓⍴DA            ⍝ Conversion pattern
[10]     R←(¯1↓⍴DA)⍴R RTA,DA                           ⍝ Convert and form array


[⍝]∇ FO.3  ⍴: 13  1986-10-22 14.45.44
[ 0]     R←LE FO DA;⎕IO
[ 1]    ⍝ FN: Convert APL2 numerics to S/370 floating point (full/double word)
[ 2]    ⍝ Extension: Function will allow full word floating point
[ 3]    ⍝ LE - Numeric scalar                         Number of bytes per element
[ 4]    ⍝ DA - Numeric array                          APL2 data to convert
[ 5]    ⍝ R  - Character array                        S/370 representation
[ 6]     ⎕IO←1
[ 7]     →(2=⎕NC 'LE')/dy                           ⍝ Dyadic call
[ 8]     LE←8                                       ⍝ Default double-word
[ 9]    dy:⎕ES(~(LE∊4 8)∧0=≡LE←''⍴LE)/5 4          ⍝ LE scalar and LE∊4 8
[10]     →(3 11 ⎕NA 'ATR')/ok                       ⍝ Check/create association
[11]     ⎕ES 1 2                                    ⍝ Unable to associate
[12]    ok:R←((⍴DA),LE)⍴('(* E5 *)'⍕LE)ATR DA     ⍝ Convert and form array


[⍝]∇ II.3  ⍴: 12  1987-02-19 13.05.27
[ 0]     R←II DA;ND;⎕IO
[ 1]    ⍝ FN: Convert S/370 integer data to APL integer array
[ 2]    ⍝ Restriction: Will only accept 2 or 4 byte integers.
[ 3]    ⍝ DA - Character array                          Integers to be converted
[ 4]    ⍝ R  - Numeric array                            APL integer array
[ 5]     ⎕IO←1
[ 6]     ⎕ES(~(≡DA)∊0 1)/5 4                          ⍝ Error not simple
[ 7]     ⎕ES(~(ND←¯1↑1,⍴DA)∊2 4)/5 4                  ⍝ Error not 1-7 bytes
[ 8]     →(3 11 ⎕NA 'RTA')/ok                         ⍝ Create/check association
[ 9]     ⎕ES 1 2                                      ⍝ Unable to associate
[10]    ok:R←'(I5 9 5555559)'⍕ND,1,×/¯1↓⍴DA           ⍝ Conversion pattern
[11]     R←(¯1↓⍴DA)⍴R RTA,DA                          ⍝ Convert and form array


[⍝]∇ IO.3  ⍴: 12  1986-10-22 14.45.44
[ 0]     R←LE IO DA;⎕IO
[ 1]    ⍝ FN: Convert APL2 integers to S/370 representation of integers
[ 2]    ⍝ Restriction: Output length is limited to 2 or 4 byte integers
[ 3]    ⍝ LE - Numeric scalar                         Bytes per output integer
[ 4]    ⍝ DA - Numeric array                          Data to be converted
[ 5]    ⍝ R  - Character array                        S/370 representation
[ 6]     ⎕IO←1
[ 7]     ⎕ES(~(LE∊2 4)∧0=≡LE←''⍴LE)/5 4             ⍝ LE scalar and LE∊2 4
[ 8]     ⎕ES(∨/,DA≠⌊DA)/5 4                         ⍝ Error, not integer
[ 9]     →(3 11 ⎕NA 'ATR')/ok                       ⍝ Create/check association
[10]     ⎕ES 1 2                                    ⍝ Unable to associate
[11]    ok:R←((⍴DA),LE)⍴('(* I5 *)'⍕LE)ATR DA     ⍝ Convert and form array
```

```
[⍝]∇ LI.3  ⍴: 10  1986-10-22 14.45.44
[0]     R←LI DA;⎕IO
[1]    ⍝ FN: Convert system/370 bit data into APL2 boolean array
[2]    ⍝ DA - Character array                      Bit data to be converted
[3]    ⍝ R  - Boolean array                        APL boolean array
[4]     ⎕IO←1
[5]     ⎕ES(~(≡DA)∊0 1)/5 4                      ⍝ Not simple
[6]     →(3 11 ⎕NA 'RTA')/ok                     ⍝ Create/check association
[7]     ⎕ES 1 2                                  ⍝ Unable to associate
[8]    ok:R←'(B5 9 5555559)'⍕1 1,8××/⍴DA         ⍝ Conversion pattern
[9]     R←((¯1↓⍴DA),8×¯1↑⍴DA)⍴R RTA,DA           ⍝ Convert and form array

[⍝]∇ LO.3  ⍴: 11  1986-10-22 14.45.44
[ 0]     R←LO DA;⎕IO
[ 1]    ⍝ FN: Convert from APL2 boolean array to S/370 bit data.
[ 2]    ⍝ DA - Boolean array                          Data to be converted
[ 3]    ⍝ R  - Character array                        Output S/370 bit data
[ 4]     ⎕IO←1
[ 5]     ⎕ES(~(≡DA)∊0 1)/5 4                        ⍝ Not a simple array
[ 6]     ⎕ES(R≠⌊R←(¯1↑1,⍴DA)÷8)/5 4                 ⍝ Check last dimension
[ 7]     →(3 11 ⎕NA 'ATR')/ok                       ⍝ Create/check association
[ 8]     ⎕ES 1 2                                    ⍝ Unable to associate
[ 9]    ok:R←'(B5 9 5555559)'⍕1 1,×/⍴DA             ⍝ Conversion pattern
[10]     R←((¯1↓⍴DA),(¯1↑⍴DA)÷8)⍴R ATR,DA           ⍝ Convert and form array


[⍝]∇ PDI.3  ⍴: 11  1987-02-19 13.04.36
[ 0]     R←PDI DA;ND;⎕IO
[ 1]    ⍝ FN: Convert S/370 packed decimal format to APL numerics
[ 2]    ⍝ DA - Character array                       Packed data to be converted
[ 3]    ⍝ R  - Numeric array                         APL numeric data
[ 4]     ⎕IO←1
[ 5]     ⎕ES(~(≡DA)∊0 1)/5 4                       ⍝ Error not simple.
[ 6]     ⎕ES(~(ND←¯1↑1,⍴DA)∊⍳16)/5 4               ⍝ Error not 1-16 bytes.
[ 7]     →(3 11 ⎕NA 'RTA')/ok                      ⍝ Create/check association
[ 8]     ⎕ES 1 2                                   ⍝ Unable to associate
[ 9]    ok:R←'(P35 9 5555559)'⍕ND,1,×/¯1↓⍴DA       ⍝ Conversion pattern
[10]     R←(¯1↓⍴DA)⍴R RTA,DA                       ⍝ Convert and form array

[⍝]∇ PDO.3  ⍴: 11  1986-10-22 14.45.44
[ 0]     R←LE PDO DA;⎕IO
[ 1]    ⍝ FN: Convert APL numeric array to S/370 packed decimal format
[ 2]    ⍝ LE - Numeric scalar                       Number of bytes per element
[ 3]    ⍝ DA - Integer array                        Data array to be converted
[ 4]    ⍝ R  - Character array                      Packed decimal representation
[ 5]     ⎕IO←1
[ 6]     ⎕ES(~(LE∊⍳16)∧0=≡LE←''⍴LE)/5 4           ⍝ LE scalar and 1≤LE≤16
[ 7]     ⎕ES(∨/,DA≠⌊DA)/5 4                       ⍝ DA must be integer
[ 8]     →(3 11 ⎕NA 'ATR')/ok                     ⍝ Create/check association
[ 9]     ⎕ES 1 2                                  ⍝ Unable to associate
[10]    ok:R←((⍴DA),LE)⍴('(* P35 *)'⍕LE)ATR DA ⍝ Convert and form array
```

Diary of an implementer

by Paul Chapman

(Editor's note: This article comprises various thoughts recorded during the initial stages of the development of the I-APL interpreter. They have deliberately been left un-edited, in order to give readers some insight into the issues that become uppermost in the mind at various stages of such a project. I hope that Paul can be coerced into producing a more leisurely retrospective overview of the project when it has been completed.)

Monday 10th November 1986 00:10

I've spent a week getting started on the development language and environment for implementing I-APL. I wish I had six months instead of three weeks to finish this part of the project alone – it's proving possibly more interesting than the APL itself. Romilly Cocking told me that at the moment the fight is on for the world's sixth most popular language (how do they judge these things?) between FORTH, LISP and APL. Perhaps DE (for that is what this new language is called at the moment – I started by calling it DEL, but typing DEL to a DOS prompt causes other things to happen. Not too expensive, that mistake. . .), perhaps DE might be up there somewhere in a couple of years.

Some compromises have been forced, however, because of lack of space on the target machines for the APL itself. For example, it is important to know exactly how much stack space is going to be used by I-APL (porters shouldn't have the responsibility of having to put in stack overflow checks and handlers), so it must be possible to analyse completely (by machine, of course) the full calling structure of the interpreter. This means no recursion, and that is a decision I may live to regret.

Recursion is easy in a language like APL, since the workspace has a fully flexible heap manager, but I can't afford such luxuries (I must take pity on the poor porter and not give him too much original code to write). Languages like C are recursive, but then the system goes BANG when the stack overflows, and anyway, starting with 20K too much on the stack is not much of a penalty in a 512K machine.

This complete analysability also means function pointer variables are out, since they hide the structure: you may know when they're stored, but how do you know how full the stack is when the function being pointed to is actually executed? After six months of the freedom of C, these jackets will be severely straight.

Still, it means that the structure of the language, which is sort of FORTH-like but with local variables (which, as I've rediscovered from my FORTRAN days, don't have to be on the stack) and nicer looking blocks (surrounded by familiar pairs of braces, with the occasional semicolon) and some powerful self checking abilities (to reduce programmer-error – yes, it does exist, but don't tell the guys who are paying me. . .), anyway, it makes the structure a lot simpler.

I've actually got the compiler working. A week ago I was phoning up Geoffrey Roughton (the sugar daddy of VIZ::APL) asking if he had any compiler-compilers for the PC, and generally panicking as I tried to remember about LR(2) and recursive descent, which only analyse the syntax anyway and are a far cry from actual code generation. But everything came out OK in the end (thanks to recursion – oh dear, what shall I do without it?), but now the symbol-table structure is groaning under the pressure of putting local variables in (into DE, not APL – I haven't started the APL yet, but don't tell the guys who are paying me. . .), and now I wonder if a general namespace approach is possible (which makes DE itself much more saleable in the end), and how many pointers and two-way linked lists are needed and whether there is an elegant solution.

Elegant solutions are satisfying, of course, but they are also maintainable. Note well.

About four o'clock yesterday morning I discovered that my rationalised algorithms for compiling the likes of IF, FOR and WHILE structures were the same as those implemented in MVP FORTH. I know I'm on the right track.

Only three more weeks to finish DE. Yet to go in are: the DE interpreter, the DE assembler (which produces the final target code from the source), the DE debug/trace facilities (all full screen and interactive), the virtual memory manager (if I need it – I hope not. Maybe I'll buy another 128K for this machine). And I move house next weekend. I'll have to work 12 or 16 hours a day to get it done. Please tell the guys who are paying me. . .

Monday 17th November 1986 16:44

I had thought of putting the development environment up under Microsoft Windows. But then I saw the size of Windows. There simply wasn't going to be enough room, and learning the Windows system calls and so on would have taken far too long. And my budget doesn't stretch to a mouse at the moment.

Anyway, why use a pre-existing package when you can re-invent the wheel yourself? Most consultants would tell you that there is every reason, and despise the "not invented here" attitude. But when you invent something yourself, you don't have to learn how to use it. And it doesn't come overloaded with features you'll never use, which just use up space in memory and in the manuals. And if you don't like the system limitations, you can modify them. And if, after a while, it turns out that your system doesn't quite do what you had hoped, why, you can start again from scratch.

And when you've finished it, you can cobble together some obscure documentation, think up a thousand and one new uses, and then sell it at a premium to all those people who took their consultants' advice.

So I wrote a windows system. It runs on a text screen, has exactly six windows (always open), has a user editor for changing the relative size of the windows on the screen, and is small and fast.

Now I have a prototype front end, which allows me to define tables (ROM resident fixed lists of addresses and/or constants), strings (zero terminated sequences of characters), data areas (uninitialised RAM vectors), constants (numbers or characters), and functions. Functions and tables can have tables, strings, data areas, constants and functions included explicitly in-line, or, in the case of functions, in the header by name.

After I wrote the last entry, I spent some more time rationalising the workings of block structures. Here is an extract from the C source of the definitions which govern their behaviour:

```
{"case", 'b', (int) "{ocPomDpjs :xfgUpxsjflomDpjs }xfgUpxsjflocH"},
{"do", 'b', (int) "{ocN }"},
{"for", 'b', (int) "{ocPomDpjsxfmUpxsjflpjs }jfvgD"},
{"if", 'b', (int) "{ocFpjs :jfqxfgUpxsjflojs }jfl"},
{"repeat", 'b', (int) "{ocNpjs }jfvgU"},
{"until", 'b', (int) "{ocNpjs }jfvgF"},
{"while", 'b', (int) "{ocNpjs }jfvgT"},
```

(Editor: I hope that little lot has not been mysteriously translated in one of the many electronic transfers between Paul's diskette and the typesetting equipment!)

A little bit of code in the compiler just interprets the one letter instructions. The capitals are codes for actual intermediate code instructions to be compiled into the object. The '{'s, '}'s and ':'s just label the pieces to be interpreted when those characters are encountered in the source. You can see I've been having fun.

Now the operation of the user interface is in the process of being designed. Golly, how I hate designing things – APL programmers will know what I mean: with APL you just keep playing around until you like it, at no extra cost. (However, I would not recommend this approach for designing files and file based systems.)

At the moment, I'm using a word processor and program editor called WordVision, in which the function keys in the first column have global meanings (extracts, search/replace, format, files and print), and the second column have meanings depending on which of these five modes, plus one more (text entry), the operator is in. This is simple. I will try to do the same. The trouble is finding five (or six) functional headings which are logical, consistent and cover all options which are needed now or might be developed in the future.

And I still can't actually execute any of the code I type in, though I can save it on disk. By next week, however. . .

Tuesday 25th November 1986 04:05

By next week, I still can't execute the code. But all the C to execute code is written now, and all I need is a front end to get at it.

Talking of front ends, it is mostly finished now. It uses nearly every finite combination of function keys to provide facilities to edit source, examine source by token, examine object by word, list names of all objects of a particular type – constants, arrays (which used to be called data areas), functions and lists (which used to be called tables) – and so on. I can even find out how much space I've got left. Needless to say, the structure is logical, consistent and covers all the options I've thought of so far.

Dave Ziemann and Anthony and Sylvia Camacho were here on Sunday to talk about the schedule. Naively, I gave them an idea of what I thought mine might be. It is rather daunting to look at the amount of work to be done.

In the end, it all comes back to design. It seems to me that the best thing to do when you're tight for time is leave the design until as late as possible. Again, this sounds like rather the wrong advice for consultants to give, but it is possible to work this way. You just make sure that every element of the system is as general and as self-contained as possible.

Dave Ziemann lent me a book called *Reliable Software through Composite Design* by an IBMer called Glenford J Myers (would IBM ever take me on with a dull name like Paul Chapman?). He's got some crazy ideas about Module Strength and Module Coupling which are wonderful in an ideal world filled with ideal programming languages, and aren't too bad in the real world. He stresses these ideas with a view to making systems more maintainable, and they do indeed work, so I hear.

So we have the traditional development strategy: firstly, specify the problem; then design the solution, then implement it using Myers ideas, then maintain and extend it as necessary.

My approach is to regard the implementation phase as just another maintenance exercise, which means the design of the solution can be skimped a bit. To me, the absolute top-level design (what it does) is fairly obvious, and the absolute bottom-level functionality (how all the bits work) should be made up of units which are as general as possible. Then the middle level (how it does it) is just (inevitably, in my experience) the rules which best suit the requirements today, which everyone knows are different from the requirements tomorrow (or usually yesterday, because nobody's bothered to tell you yet).

Enough of these ramblings. Goodnight.

Wednesday 27th November 1986 05:00

Just a quick note to say WOW! not only is 1 plus 1 equal to 2, but 2 to the sixth power is 64. I can execute things. Now all I've got left is local variables and a few more primitives and user I/O and breakpoints and source level trace and object level trace and. . .

Monday 1st December 1986 07:15

In principle, at least, the deadline for completing the first phase of the implementation of the APL interpreter passed a little over seven hours ago. However, I measure the passing of days by going to bed, and I haven't done that yet.

It is finished. In this last session, breakpoints finally went in, as did checkpoints (show current state without stopping), and the facility to record on file all changes to any source code throughout an entire session. One important aspect of the system is that source code is stored in a straightforward text file, so that it can be examined and edited off line, i.e. outside DE itself.

It was only in the last day that I actually checked that garbage collection worked, which it did without trouble. There are still one or two features that I would like to have, but I shall strictly limit any further enhancements to DE to one hour a day. I would like to be able to change the contents of arrays without having to write a one-off function to do it, and for that matter I have to write a function every time I want to execute anything – at the moment there is no equivalent of immediate execution mode in DE.

I have started looking at the APL standard. What a nightmare. It looks like it might be possible to implement each definition and evaluation sequence as a subroutine, and then it would all work by the end. What's wrong with this approach, then? Well, you don't know whether it works until it's finished, and even if I correctly interpreted and implemented each aspect of the standard definition, which is in the form of a formal model of a particular implementation of "Standard APL", I doubt that the standard itself is without error. Perhaps a good control for the standards committee would be to employ a team of programmers who knew nothing about APL to implement the language using just the standard and without reference to any existing APL system or document. It couldn't be done, of course.

One sad fact about the standard is that it abandons the beauty and simplicity of APL constructs and algebra in favour of straight mathematical concepts such as sets, counting numbers, dense sets, open and closed intervals, and so on. I am lucky enough to have received a formal education which covered these ideas, although I find them difficult now.

I sympathise with the committee's desire to avoid having to refer to any existing APL, or to have to use a lot of circular definitions. However, it seems absurd that APL itself, which was originally developed both to express mathematical ideas in a more accessible way than that available with traditional notation, and to describe the behaviour of computers (particularly the IBM System/360), cannot be used to express the concepts and algorithms which underlie the language. Ah, well. At least once the APL standard is published, all future standards in other fields will be able to use the APL notation to clarify and simplify.

So now my first task is to get the free storage (heap) management subsystem working, and then to concentrate on the design of fundamental things like the symbol table, internal storage of arrays, the state indicator, and the APL tokeniser. Maybe by next week I will have found that DE is totally unusable. I hope not.

Tuesday 9th December 1986 04:40

Well, not bad. I have the free storage manager running, and the symbol table, and two different versions of the lexical analyser – one based on a code-driven version of the actual ISO standard diagrams, and one based on a finite state machine (FSM).

The first works fine, but is a little slow, since it has to try to fit the arriving characters to various patterns, which means the character '+' would be attacked as if it were the first character of a numeric vector, a name, or a character constant before the software realises that it must be a primitive.

So I tried the FSM approach. Backtracking is more difficult here, and must be sidestepped by using more states. I have 15 at the moment, and it looks as though I might end up with many more if I'm not careful.

The problem lies in the fact that the characters '.' and overbar (high minus), as well as having meaning in numeric constants, are also designated as "primitives" for some bizarre reason. So have a look at this:

1 1.1E1	parses as you would expect as a two element vector;
1 1.E1	too; and then not surprisingly
1 1.1E	parses as a syntax error; but, hold on
1 1.E	as the vector 1 1, the inner product operator '.', and the identifier E.

Now, as far as it goes, this doesn't matter too much, because inner product isn't defined in the standard for array or user defined arguments. But from the treatment of characters like '.', overbar and underbar as having at least lexical meaning in this version of the standard, one can deduce with reasonable confidence that extensions are envisaged that would also give meaning to these characters when they stand alone. In some visions of APL, they already have meaning ('.' for the generalised inner product, where 1 1 . E is defined semantically at least, and overbar and underbar have been used for plus/minus infinity).

Now, the standard insists that a conforming APL system must behave in a consistent manner on a conforming program. Consider this:

```
1 1.E FOO 2
```

According to the standard, this statement will be successfully decomposed into "lexical units" (as explained above), and the part-expression "E FOO 2" must be executed before the erroneous '.' is found. Nowhere in the standard does it specify that a conforming program must run without error, although one part of the definition of a conforming program does relate to the signalling of errors. I shall come back to this in a moment.

So, FOO itself is called and executed before the inner product operator is parsed and the error is detected. I think I would prefer to be told that "1.E" is a badly-formed number before FOO destroys my data base (E contains 2 0, and of course you realise that FOO makes the tea if the first element of the left argument is a 1 (as intended), and deletes the data base if it is a 2 (as it turned out)).

So now here are the problems. There are two, one for me (the implementor striving to produce a standard conforming APL), and one for ISO (the international organisation attempting to standardise the implementation and use of APL).

Mine is that I can't figure out if I am allowed to decide that 1.E is a syntax error before I execute FOO. On the one hand, this is specifying behaviour different from that required by a literal interpretation of the standard, and is certainly not a consistent extension, which may only be made by removing an error signal, not by inserting one. On the other hand, to quote the standard: "A conforming-program shall not depend on the signalling of any error by a conforming-implementation." The question arises, is my program conforming? Does it rely on the signalling of an error by a conforming implementation?

Well, that depends on whether it is possible to build a conforming implementation which produces a different left argument for FOO. Suppose we have an implementation which interprets "1.E" as "1.E0". This would certainly change the behaviour of the program, but is it conforming? It removes the syntax error which would have been reported after FOO was executed, so in that sense it is a consistent extension. But it changes the way the lexical analyser works, although no error is reported by either version at the lexical analysis stage.

Finally, if this new interpreter is standard conforming, then my program is not. If the program is conforming, then the interpreter is not. But who is to say which way round?

Now, if you're still with me, what about this?

```
1.J2.
```

Again, according to the standard, the lexical analyser has no problem with this. It is the number "1", the inner product operator '.', the identifier "J2", and the inner product operator again. Of course, there is no semantic analysis possible within the standard.

So what if I want to build an APL interpreter which uses this well-known syntax for complex numbers? Can I write a standard conforming interpreter? It definitely removes an error, at the semantic analysis stage. But it alters the behaviour of the lexical analyser significantly.

So now you are as confused as I am.

Now for the ISO. The first part of their problem is the very specific interpretation of the behaviour of the lexical analyser. This could be easily fixed (except that it is too late for that. . .), either by insisting on one or more decimal digits after a decimal point in a number (1.1J2.2 produces a syntax error during lexical analysis according to the standard, so a consistent extension is obviously possible), or insisting that the '.' (and preferably also the overbar and underbar), when not used in a numeric constant, be separated from any letter, digit, '.', overbar or underbar by at least one space (as, for example, numbers must be separated from letters: 123ABC is a syntax error according to the standard).

The second part of their problem is catastrophic. The standard goes to great lengths to define the terms it uses precisely, and prints such defined terms in bold letters. Forward references are avoided where possible, and are usually pointed out where they do occur. It seems to me that the definitions of conforming-implementation and conforming-program are central if the purpose of the standard is to assist interpreter writers and APL programmers to produce conforming implementations and programs, and to assist the ISO in determining whether implementations and programs submitted for approval are in fact standard conforming.

So here are some extracts. Part of the definition of conforming-implementation is this:

"A conforming-implementation shall provide all defined-facilities and implementation-defined-facilities. Each such facility shall behave as specified by this standard."

The words "defined-facility", "implementation-defined-facility" and "facility" are defined terms. Here is part of their definition:

"Facility (of an implementation): A unit of behaviour. Every facility is one of:

"Defined-Facility: A facility fully specified in this standard and not designated optional or implementation-defined.

"Optional-Facility: A facility fully specified in this standard and designated optional.

"Implementation-Defined-Facility: A facility not fully specified by this standard that is designated implementation-defined."

Now there are specific places in the standard which refer to the terms optional facility and implementation-defined-facility (eg the shared-variable-protocol is defined as an optional-facility). But nowhere is any algorithm defined as a defined-facility (at least, the term has only three references in the index: its definition, its use in the definition of conforming-implementation, and its appearance in the notes accompanying the definition of the shared-variable protocol, pointing out, tautologically, that this optional-facility is not a defined-facility). In this sense, the terms facility and defined-facility are defined in terms of each other (hee hee !). To make matters worse, section 16.2.1 is called "User Facilities" and the word "facility" is used freely throughout the section, presumably standing in this instance for itself since it does not appear in bold type.

So what is a defined-facility? If everything is, then everything has to be implemented literally. But the symbol-table is defined as "A list of all symbols whose names are unique", which, if interpreted literally, would require, for a maximum symbol length of 6, in excess of 27 gigabytes to store! So they can't mean that. Which calls into question the specific behaviour of the lexical analyser. If it is a "defined-facility", must it be implemented literally? Can the lexical analyser and syntactic analyser and semantic analyser be taken together as a single defined-facility, in which case, as long as an error is eventually produced for "1.E", does the error have to be signalled in the lexical analyser?

The standard doesn't say, and I have tried to demonstrate that there is no common-sense solution to these difficulties of interpretation.

Well, there is one. Ignore the standard, and do the sensible thing. But what will the ISO do then? Of course, if everybody's idea of the sensible thing is the same, there is no problem. The ISO can adopt this sensible interpretation of the standard, and verify implementations on that basis. But IBM, IPSA, STSC et al all believe that they are each pretty sensible, and look where they're leading us...

Wednesday 17th December 1986 01:56

The floating point representation has been decided and implemented. It is 8-digit BCD with a 7-bit binary biased exponent. This means that all arithmetic is performed internally in decimal rather than binary. This proves to take rather a long time – division in particular must be implemented as long division, which is almost as hard for computers as it is for people.

There are a few reasons why I decided to do it this way. Firstly, I had a lot of trouble when writing VIZ::APL with converting between ASCII representations of numbers and internal binary (or, in fact, hexadecimal) floating point. The only way of working out the internal format for 1E70 is to raise 10 to the 70th power (effectively – in fact, I had power of two powers of ten stored ready, so that 1E70 was calculated as 1E64 x 1E4 x 1E2). This could lead to loss of precision in the floating point routines even before the user started his calculations. Using BCD this problem is overcome.

Another reason is to be sure that 1 divided by 5 really is .2 – children expect this, and with binary representation, the calculation produces a recurring binary (like 1/3 is a recurring decimal, .3333. . .), which may not come out quite as intended.

Now, I know a lot of work has been done on this sort of thing, and it is possible to get binary floating point representations to behave in a sensible and predictable way, so that most of these sorts of problems are removed. Unfortunately, I skipped numerical analysis (for that is how the subject is known academically) during my degree, so I am still in ignorance of the techniques involved.

In fact, I have a strange superstition regarding floating point numbers – I do not quite believe that they're real (get the pun?). I very rarely program in floating point, and when faced with the inevitable (like graphics for example) I try to avoid floating point until the last minute, and even then will prefer to write my own graphics interface to address the pixels directly, rather than having to try to figure out how to scale my numbers to fit the 1024 by 1024 (or whatever) co-ordinate system used.

Also on this point, I would like to say that there seems to be a confusion in graphics between points and cells – in other words, is a pixel a point with floating point co-ordinates, or is it a finite cell which is addressed by its centre? The problem becomes important when trying to find the distance between pixels in the internal scale of the package (eg STSC APL) when you know the number of pixels. On the Hercules card there are 720 physical pixels horizontally, and this is mapped to the range 0 to 1024 in APL. If the pixels are cells, each is 1024/720 units wide. But if they are points, each is 1024/719 units wide.

In fact, both of these are wrong. The answer is 1023/719! Figure that one out. This is why I don't like floating point. . .

The last reason for choosing BCD is that I haven't done it before – it was different, and why shouldn't I have a little fun?

Anyway, it all works. I'm a little worried about speed – worse case division (99999998 / 33333333) takes 40 seconds on the development system. I anticipate about a one-hundredfold speed-up for the live version, but still .4s seems a little long. I will have to think on this further.

APL + PC's

Whatever your area of interest, if it includes APL on a PC, Mercia will have something to offer. Hardware, Software, Liveware – we're sure you could use some of our products or services to make your APL work even more productive. Take a look through this summary – then give us a ring for more details.

APL SYSTEMS SOFTWARE/HARDWARE

ALLcard –	A REAL benefit to APL*PLUS/PC users with workspace size problems. Up to 750,000K REAL workspace. Check your ⎕WSSIZE in a CLEAR WS, and think how you could use those extra bytes. If that's not enough, it does up to 10Meg battery backed RAM disks, and all the standard expanded memory specs.
Multi-APL –	Multi-user/Multi-tasking/LAN for APL*PLUS, includes APL file component locking etc.
APL*PLUS/PC –	The old favourite – Lower price, full support, plus all the add-ons.

APPLICATIONS SOFTWARE IN APL

STATGRAPHICS –	Comprehensive statistical Graphics System – PLUS our unique customisation service for users who don't think it's comprehensive enough.
Exec-U-Stat –	Stats and graphics for the business user.
LOGOL –	R G Brown's latest logistics decision support system for the PC. Forecasting, Inventory Control, Distribution, Master Scheduling.

CONSULTANCY

Could you use some extra Liveware? Mercia's consultants are experienced APL'ers, specialising in OR, stats and math's applications – both London and Birmingham based.

TRAINING

See our course listing elsewhere in Vector, if none suits please contact us regarding In-house and specialised courses.

MERCIA SOFTWARE LIMITED

Aston Science Park, Love Lane, Birmingham B7 4BJ.
Telephone: 021-359 5096

Public Domain Software Library
Call for Software

Purpose

The purpose of the British APL Association Public Domain Software Library (PDSL) is to provide useful software to the APL community. In order to achieve a high level of service, we need a good supply of quality software. During the initial phase of the library, we are therefore advertising for submissions from everyone in the APL community.

What kind of software?

We are looking for anything that could be of interest to the APL programmer, user or manager. It might be a tool for migrating APL workspaces between different interpreters, general APL utilities, APL2 functions for simulating component files, a reverse Polish model of APL or even a DOS adventure game. Each of these examples is an actual disk from the library. There are no restrictions on the type of software, type of APL or the type of machine on which the software is to be run. As long as it is of interest to APLers, then it is worthy of consideration. The programs do not even have to be APL, although we have no desire to flood the PDSL with non-APL software available elsewhere. Public domain software should normally include 'unlocked' source code wherever possible.

Transfer medium

The transfer medium for the BAA PDSL is the DOS-format five-and-a-quarter inch floppy diskette. If you can get it onto a floppy disk, then we can make it available to others. If the target machine is not a PC, then appropriate transfer instructions should be included on the disk.

Documentation

The library will not currently handle any medium other than the floppy disk. In particular, we do not undertake to distribute paper documentation of any kind. All disks must therefore include sufficient user documentation to permit effective use of the software. The donor may include an address where paper documentation can be obtained, possibly at a nominal cost.

Charges

It is not intended that the BAA PDSL make a profit. Our charging policy is designed to cover the direct costs of materials, postage and handling, although we do make a reduced charge to BAA members in order to encourage membership.

A PDSL software donor must not require payment from the user, although reasonable charges for 'registration' or paper documentation may be advertised on the disk. Disks that demonstrate commercially available products are acceptable, provided that this is made clear.

Liability

The donor's signature is required to declare the donor's right to allow the BAA to copy the software, and to permit anyone to copy and use it. Software is accepted by the BAA on good faith, and we do not vouch for or make any claims regarding donated software. The BAA cannot be held responsible or liable for any damage, however caused, by the use or misuse of library software.

If you are considering creating software for the library, please read the software submission form first, so that you know what is expected of you. Please share the fruits of your work with others.

British APL Association Public Domain Software Library
SOFTWARE SUBMISSION FORM - Page 1 of 2

Please copy and fill in this form for EACH disk you submit.

Details corresponding to items flagged (*) will not be made publically available, but are for our records only.

Use BLOCK CAPITALS for all items except numbers 6 and 16.

0. Submission date:__________

1. Name of donor:____________________ 2. Daytime phone(*):__________

3. FULL address(*):____

____

4. Electronic mail addresses(*):__________________________________

5. Disk title:____

6. Brief description of disk contents:____________________________

____

____

____

7. List target machine:____

8. List additional software required: ____________________________

9. Indicate special hardware requirements:_________________________

10. Is user documentation provided on the disk?(Y/N):_______________

11. List titles of any paper documentation included with your submission:

____

12. Is this documentation, or any other, available to users upon application to you?(Y/N):_____ Please give details:______________

____

13. Does the disk include any form of payment request from users of the software?(Y/N):_____ Please give details:_______________________

____

14. Does your submission constitute a 'demonstration disk' in that it demonstrates software that is available for purchase?(Y/N):_________

15. If the TARGET machine is NOT a DOS-based PC, does the disk include instructions for transfer to the target machine?(Y/N):____________

British APL Association Public Domain Software Library
SOFTWARE SUBMISSION FORM - Page 2 of 2

16. File names and descriptions. This information will be made publically available in the software library catalogue. Please save us some work by including these details on a file named <CAT> (no file extension) on your submitted disk. Please enter these details for all files on the disk, except <CAT> itself. You can affix a print of <CAT> below.

For APL workspaces, please use the space below to document functions.

FILENAME.EXT	SHORT DESCRIPTION

17. Please use extra space above for any comments you wish to appear in the library catalogue.

18. Your signature is necessary. It declares your legal right to make the disk freely available for copying and use, and grants the British APL Association a similar right.

Signature:____

Mail your submission to: The BAA Public Domain Software Library
c/o Flat 3, 63 Queens Crescent
London NW5 4ES
ENGLAND

NEW MAINFRAME SOFTWARE FOR IBM'S® APL

Now available in the UK, two new offerings from STSC that enhance IBM's mainframe APL implementations

If you're staying with VS APL ...

COMPILER The first commercial compiler for APL compiles functions individually. Results in significantly faster execution. Interpreted functions can call compiled functions and vice versa.

If you're migrating to APL2 ...

SHAREFILE/AP STSC's popular APL component file system is now available under APL2. Multi-user, nested array storage, libraries, access matrices. Multiple file system support. International language translations.

For full information, contact the APL*PLUS™ Product Group, Cocking & Drury on 01-493 6172.

Trademarks/Owners: IBM/International Business Machines Corporation · APL*PLUS/STSC, Inc.

COCKING & DRURY LTD.
THE APL PROFESSIONALS

16 BERKELEY STREET · LONDON · WIX 5AE
Tel: (01)493 6172 · Tlx: 23152 MONREF G

Index to Advertisers

Ampere	6
APL People	93
APL Software Ltd	101
Cocking & Drury	8, 122, 142
Dyadic Systems Ltd	80, 81
HMW Programming Consultants	116
IBM (Sweden)	54
Mercia Software	138
MicroAPL	2
Peter Cyriax Systems	30
Vector – Back Numbers	42

All queries regarding advertising in VECTOR should be made to the advertising editor, Cathy Dargue, at the following address:

Cathy Dargue,
60 Downhall Ley,
Buntingford,
Herts SG9 9TL.
Tel: 0707–32516

Advertisements should be submitted in typeset, camera-ready A5 portrait format with a 20mm blank border. Illustrations should be black-and-white photographs or line drawings. Rates are £250 per page. A6 and A7 sizes (at £150 and £75 respectively) are available, subject to layout constraints.

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 – 30th April.

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):	86/87	
Non-voting student membership (UK only)	£ 5	
UK private membership	£ 10	
Overseas private membership	£ 18	$ 27
Airmail supplement (not needed for Europe)	£ 8	$ 12
Corporate membership	£ 85	
Corporate membership Overseas	£140	$210
Sustaining membership	£360	

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. It is administered by a Committee of 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.

1986/87 Committee

Chairman:	Dick Bowman 01-634 7639	CEGB, 85 Park Street, London, SE1.
Secretary	Anthony Camacho 0727-60130	2 Blenheim Road, St. Albans, Herts., AL1 4NR.
Treasurer	Mel Chapman 0785-53511	12 Garden Street, Stafford, ST17 4BT.
Activities	Phil Goacher 03727-21282	27 Downs Way, Epsom, Surrey.
Journal Editor	David Preedy 01-541 1696	Metapraxis Ltd., Hanover House, Coombe Road, Kingston, KT2 7AH.
Education	Norman Thomson 0962-54433	IBM Mail Point 188, Hursley Park, Winchester, Hants., SO21 2JN.
Publicity	Bernadette Leverton 01-622 0395	MicroAPL Ltd., Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London, SW8 5BP..
Technical	Dave Ziemann 01-493 6172	Cocking & Drury Ltd., 16 Berkeley Street, London, W1X 5AE.
Recruitment	Christine McCree 01-560 5151 (ext 3414)	Beecham UKCS, Beecham House L/2, Great West Road, Brentford, TW8 9BD.
Projects	David Eastwood 01-622-0395	MicroAPL Ltd., Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London, SW8 5BP.

Journal Working Group

Jonathan Barman	0374-588835
Anthony Camacho	0727-60130
Cathy Dargue	0707-32516
Val Lusmore	0225-62602
David Preedy	01-541 1696
Adrian Smith	0904-53071
David Ziemann	01-493 6172

Activities Working Group

Philip Goacher	03727-21282
Maurice Jordan	01-562 3090
David Parker	01-834 2333

WP disks converted to photosetting by Capella House, Stowmarket (0449) 677663.
Printed in England by Short-Run Press Ltd., Exeter.

VECTOR

VECTOR is the quarterly Journal of the British APL Association and is distributed to Association members in the UK and overseas. The British APL Association is a Specialist Group of the British Computer Society. APL stands for "A Programming Language" – an interactive computer 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. 0225-62602
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.	Park House, The High Street, Alton, Hants. Tel. 0420 87024
H.M.W. Programming Consultants Ltd	142 Feltham Hill Road, Ashford, Middlesex, TW15 1HN Tel. 07842-41232
Inner Product Ltd.	Eagle House, 73 Clapham Common Southside, London SW4 9DG. Tel. 01-673 3354
Mercia Software Ltd.	Aston Science Park, Love Lane, Birmingham, B7 4BJ. Tel. 021-359 5096
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, SW1H 0DX Tel. 01-222 7033
Peter Cyriax Systems	213 Goldhurst Terrace, London, NW6 3ER Tel. 01-624 7013 (Answerphone) T.B.A. (Mobile)

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