VECTOR

APL in Education: Special Issue

- I-APL (Report and review)
- Teaching Statistics with APL
- APL: a Lecture for Teachers
- Competition Result
- Public Domain Software Catalogue

The Journal of the British APL Association

A Specialist Group of the British Computer Society

Vol.4 No.3 January 1988

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.

CONTENTS

		Page
EDITORIAL: Computers in Mathematics	Norman Thomson	3
APL NEWS		
Quick Reference Diary	Adrian Smith	5
APL Course Diary	Cathy Dargue	7
General Correspondence	Camacho et al	8
British APL Association News		
Details of EGM and Postal Ballot	Graham Parkhouse	10
News from Sustaining Members	Cathy Dargue	11
The Education VECTOR	Norman Thomson	18
REVIEWS		
APL Booklist		20
APL Product Guide	Cathy Dargue	21
An Introduction to APL for the IBM PC & XT by Murray and Pappas	Anthony Camacho	32
I-APL for the IBM PC	Dave Weatherby	37
APL*PLUS Version 7.0	Adrian Smith	43
RECENT MEETINGS		
Under the bonnet - Paul Chapman	Report by Eileen Wilks	48
From Mainframes to Smaller Machines	Paul Smith	57
Origins	Graham Parkhouse	67
GENERAL ARTICLES		
Teaching Statistics in University College Swansea	A. D. Mayer & A. M. Sykes	74
I-APL: History and Achievements	Anthony Camacho	89
A Demonstration of Direct Definition	Anthony Camacho	95
TECHNICAL SECTION		
Technical Editorial	Jonathan Barman	102
Hacker's Corner (1)	Adrian Smith	103
Technical Correspondence	Thomson, Small	104
Prize Competition .... Result	Derek Wilson	109
A Note on Weighted Least Squares	Kelly and Thomson	115
Interfacing to PC Assembler (II)	P Wortham (UNIWARE)	119
Public Domain Software Library		
Software submission details	David Ziemann	124
Software order form	David Ziemann	126
Index to Advertisers		

dyalog APL

UNLEASH THE POWER OF YOUR IBM PS/2 MODEL 80, COMPAQ 386, OR COMPATIBLE WITH DYALOG APL/386

Now you can develop serious large-scale APL applications on a PC because Dyalog APL/386 is FAST (many times faster than APL*PLUS/PC) UNRESTRICTED (no practical limits on workspace and object size)

and gives you

▼

Nested arrays, defined operators, etc.

▼

Multiple concurrent APL sessions with 'hot key' facility

▼

Powerful CGI graphics interface

▼

Concurrent UNIX/DOS support

▼

Multi-user environment

▼
▼

▶ ▶ ▶▶ **From only £995** ◀◀ ◀ ◀

Dyalog APL/386 runs under Xenix on the IBM PS/2 Model 80, and on a wide range of 386 PC-AT compatible systems including Compaq, Wyse, Zenith, Olivetti, ITT XTRA and TI. It also supports certain 386 'turbo' cards including the Orchid 386 and Intel INboard.

Evaluation kit (including Xenix) available

For further information, contact
Sales Department
Dyadic Systems Limited
Park House, The High Street, Alton,
Hampshire, GU34 1EN. United Kingdom
Telephone: (0420) 87024. Telex: 858811

IBM and IBM Personal System/2 are registered trademarks of International Business Machines Corporation
APL*PLUS is a registered trademark of STSC, Inc
UNIX is a trademark of AT&T Bell Laboratories
Xenix is a trademark of Microsoft Corporation

Guest Editorial: Computers in Mathematics

by Norman Thompson

APL is quite simply the best programming language for teaching mathematics. This is the claim which we made last July to promote the teachers' course, and which I-APL will make again to support its campaign for APL to become an everyday language in schools and colleges.

While few readers of this magazine will wish to contest the generalities of this claim, the onus of proof is on APL supporters to explain to the as yet unpersuaded mathematical world not just the fact of the claim, but its practice: that is how in the classroom and the maths lab APL can be applied to make mathematical education more effective and what are its special qualities that put it way ahead of other programming languages for this purpose.

A symposium held at Strasbourg in 1985 by the International Council for Mathematical Instruction was devoted to the influence of computers in Mathematics and its teaching. At this conference no whisper of APL was heard, a situation which strenuous efforts will be made to rectify at the next ICMI conference in Budapest this coming July.

A selection of papers from the Strasbourg conference has been published by Cambridge University Press with Professors A.G. Howson and J.P. Kahane as editors. It costs £6.50, and is recommended to anyone for whom the combination of computers and mathematics teaching is a concern. Some fascinating issues are considered, e.g. would a present day Newton or Gauss be beguiled by computers to the detriment of mankind. A real danger is that the mindless execution of computers becomes a substitute for hard thinking, and the challenge is that of providing a powerful tool, which does not become, in Professor Burkhardt's words, "a net absorber of effort and attention". Surely this is where APL has an answer in providing a resource which is as natural to mathematicians as pencil and paper.

We must not of course behave as if computers addressed all parts of the mathematical curriculum equally. On reading the above book one is struck by the emphasis on symbolic mathematics as exemplified by the packages MACSYMA and MUMATH, the latter of which is well within school budgets. Of course what APL offers is much more, but at least APL people should be sensitive to what is seen as the perceived need by the mathematical educators themselves.

Investigation and exploration are other areas in which the application of APL is to the APL enthusiast "obvious", but the practical realities of accommodating teaching style to embrace the possibilities is quite another matter - how to effect the change in "classroom dynamics" that the micro as "teaching assistant" demands.

A forthcoming Open University course (M371 - Computational Mathematics) addresses this problem for the distance learning student. The accompanying software has two components; the first uses the Amstrad micro as a tutor, and the second is a mathematics package in the accepted sense. This may well be the first attempt at such a software combination on this scale, and it certainly merits scrutiny beyond that of the immediate course population.

Ultimately computers in mathematics will only be valuable to the extent that they provide a bridge to the abstract thinking which is the true substance of mathematics. While the availability of APL in schools is to be welcomed, the real educational challenges are only now about to begin.

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

Quick Reference Diary

Date	Venue	Event
1988		
15 January	London	BAA Meeting
1-5 February	Sydney	APL88 .... APL Past, Present and Future
18 March	London	BAA Meeting
20 May	London	BAA AGM and Meeting
16 September	London	BAA Meeting
28-30 September	Canterbury	APL-Ication the Practical State of the art APL Conference
21 October	London	BAA Meeting
18 November	London	BAA Meeting

All British APL Association meetings are to be held at the Royal Over-Seas League, Park Place, near Green Park Tube station. Meetings begin at 2.00pm and close at 5.30pm.

Dates for future issues of VECTOR

	Vol.4 No.4	Vol.5 No.1	Vol.5 No.2
Copy date	20th Feb 88	1st May 88	1st Aug 88
Ad booking	7th March 88	16th May 88	22 Aug 88
Ad Copy	21st March 88	25th May 88	31st August 88
Distribution	April 88	July 88	October 88

MicroAPL for ...

APL on the widest range of micros

PCs

MultiAPL - APL.68000 under PCDOS

Sample configurations:

68000/1MB	£ 995
68000/1MB/Maths processor	£1295
68020/1MB/Maths processor	£2995
68020/4MB/Maths processor	£5495

Prices include: Plug-in card, APL.68000, Keyboard stickers, Manual, STSC emulation software

STSC's APL*PLUS £450

Single user micros

APL.68000 for the Apple Macintosh
APL.68000 for the Atari ST — All versions £86.91
APL.68000 for the Commodore Amiga

Prices include: APL.68000, Keyboard stickers, Manual, Reference card

Multi-User systems

APL.68000 runs under the following operating systems:
Unix, CP/M68K, Wicat MCS, Mirage
on computers which include:
Sun, NCR, Wicat, HP200/300, Spectrum, Aurora, Stride
Prices range from £1200 to £2000

All prices quoted exclude VAT

MICRO APL

MicroAPL Ltd.
South Bank Technopark
LONDON, SE1 6LN
Telephone: 01 922 8866

Access and Barclaycard Orders accepted

APL Training Courses

(Prices quoted are per course, unless otherwise stated)

Dates shown cover the period from January to March 1988. For confirmation of dates and/or further details please contact the vendor directly.

Level	Company	Days	Price(£)	Dates
Beginners	Cocking & Drury	3	445	29/3
	MicroAPL	1	poa	14/1,10/3
	Uniware	5	poa	call
Beginners/PC	Mercia	3	390	19/1,22/3
Intermediate	MicroAPL	1	poa	21/1,3/3
ditto/PC	Mercia	2	260	23/2
	MicroAPL	1	poa	4/2,17/3
	Cocking & Drury	3	525	1/2, 21/3
Advanced	MicroAPL	1	poa	25/2
	Uniware	5	poa	call
System Design	Cocking & Drury	4	700	29/2
	Mercia	3	420	8/3
Statgraphics	Cocking & Drury	2	280	10/2
	Mercia	1	120	25/2
	Uniware	2	poa	call
U'ware Toolkit	Uniware	2	poa	call
R.BASE 5000	Uniware	5	poa	call

The following vendors run courses, details of which may be obtained directly from the vendor:

APL People
Inner Product
M.B.T.
Parallax

General Correspondence

The VECTOR working group welcomes correspondence on any topic affecting the APL community. All letters should be addressed to the Editor, and should indicate whether they are for the general or technical section. (Letters containing APL code will normally be classed as 'technical'.) 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.

I-APL Status Report

From: Anthony Camacho 17th Dec 1987

Dear Adrian,

You will be glad to hear that at last the first version of I-APL is available. It is suitable for learning and teaching but not for commercial work. We discourage attempts to use it commercially and prohibit its sale for direct commercial gain.

We will be sending out order forms around the turn of the year. Would you allow us to include one as an insert in Vector Vol 4 issue 3? The complete package is likely to cost £11 plus post and packing. There will be a 360K disk, an I-APL/PC Instruction Manual (52pp), "An APL Tutorial" by Thomson and Alvord (48pp) and "An APL Encyclopedia" by Helzer (it is a comprehensive reference manual - 308pp). All the books will be available separately and are much cheaper than photocopies. As we encourage copying of the disk there is no need for anyone to buy more than one. All three books have gone to press and the printers promise delivery before 31 December.

The BBC version, ported by Tony Cheal, will be available in early March probably in two stages. The first will be a disk suitable for the Master or BBC B with 32K of sideways RAM. The second will be a pair of EPROMs for any BBC. There are five more ports working which will soon be under test and many others not yet ready for testing.

The next stage of the project is the really difficult one. That is to persuade the educational world to try APL out in earnest. I suggest that everyone keen on the project who has access to a PC or clone should buy a copy of the complete package and offer to demonstrate it to teachers at their local schools. Success is getting them to buy the package and ordering another one to try on the next school.

People who do this should be able to collect reactions and opinions about the product and the workspaces which accompany it. We welcome serious practical suggestions for improvement or additions to the package which your teachers would find valuable, particularly if you can produce these and ensure that the teachers are happy with them before sending them to us. We would be most grateful for any such workspaces which we could circulate on the distribution disk.

In Europe the Danish APLers are leading the way. Their distribution plans are well advanced and they have Kim Andreason's excellent port on the RC Piccoline, the Danish school computer to distribute. The Finnish, Swiss, Swedish, German and Austrian groups are working hard at ports and translations of the texts. Several groups are eagerly awaiting the ports on the CBM 64, the Sinclair/Timex Spectrum or the Apple II. The project would be very glad to hear from groups willng to work at distribution in other countries.

Now that distribution has begun we suggest that supporters of the project contribute first to their own national distribution effort. In the UK this is being handled by I-APL Ltd, but in case we are accused of selfishness I should tell you that the project will not use internationally contributed funds to pay for UK-specific distribution. We have borrowed to pay for the printing of the texts and will probably have to do so again to print the Instruction Manuals for other versions. If sales are good we will be able to repay the loan.

The UK mailing list now consists of well over 800 and about 150 of these are teachers. It costs us about £200 to send out a mailshot to everyone. I had the good news this week that SigAPL are making a further contribution to project funds to help with distribution.

We are very grateful to all our contributors and particularly to SigAPL and the British APL Association - the largest contributor of all - please don't stop now, when the light at the end of the tunnel is getting really bright.

Yours sincerely,

Anthony Camacho
2 Blenheim Road
St Albans

... and on behalf of Ed Cherlin, Norman Thomson, Howard Peelle, Dave Ziemann and Paul Chapman.

British APL Association News

Postal Ballot

At the Extraordinary General Meeting of the BAA held at 2p.m. on 20th November 1987 at the Royal Over-Seas League, Park Place, London, a resolution was passed, nem con, which amended the BAA's constitution to require the Committee to be elected by postal ballot. The new procedure will be used to elect the 1988/89 committee who will be holding office from 1 May 1988 to 30 April 1989.

The nominating committee will have produced a slate of candidates by the time this edition of VECTOR is published. Ballot papers will be issued on 1 March, with a photo and a statement from each candidate, and the result of the ballot will be announced at the AGM on 29th April. If you wish to nominate a candidate please arrange for the following information to be in Graham Parkhouse's hands on or before 12th February 1988.

1. Name, address and telephone number of nominee, and the office for which he/she is standing, with a signed statement of willingness to stand, and if elected, to execute the office diligently. The nominee must be a member of the BAA, and a nominee for Chairman must be a member of the BCS.

2. A proposer and a seconder, who must both be BAA members, should submit their nominations in writing, identifying themselves clearly.

3. A black and white passport-sized photograph of the nominee and a platform statement of not more than 250 words (400 for Chair, Secretary and Treasurer). Both these will be circulated with the ballot papers.

Send this information to Graham Parkhouse:

Hon. Sec. BAA,
Dept. of Mechanical Engineering,
University of Surrey,
GUILDFORD GU2 5XH.

News from Sustaining Members

Compiled by Cathy Dargue

APL People Ltd

As this is a special, education issue of VECTOR, there will undoubtedly be those of you reading it who will not have seen a copy of VECTOR before and possibly not have had very much to do with APL before either. So let me take this opportunity to describe what APL People is all about for those of you who do not yet know us!

First and foremost we are a Company which specialises in supplying consultancy in APL, for which we use our own staff and the services of a growing number of freelance APL consultants. (Any freelance APL consultants wishing to be included on our register should contact Val or Jill in the first instance - number on back cover of this Vector).

Another service for which we are well known within the APL world is our APL employment agency through which we place people with APL skills in permanent jobs both in the UK and abroad. I'm sure you will all be encouraged to hear that the future has never been brighter for people seeking new jobs, especially those with two to five years' APL experience. We are making a conscious effort to expand this side of the business to place more people in jobs abroad and in the States. (Please take note all you employers out there - we're waiting to hear from you!) The current demand is very much for people with APL2 experience, and we are starting to be asked more and more whether we have any people who are familiar with C in addition to APL.

And finally, we also develop and market a range of software written in APL. This includes RDS, a relational database system, now also available on the PC, and our Production Engineering packages, PEFAC and MANTRAC. PEFAC, the multi-user estimating package, now interfaces with other Production Engineering packages written in Basic, and will be available running under Unix in the Spring.

Dyadic

Dyadic is pleased to announce the availability of Dyalog APL/386, a new version of Dyalog APL for the IBM PS/2 model 80, Compaq 386 and other Intel 80386-based computers.

Dyalog APL/386 runs on the IBM PS/2 Model 80; and on a wide range of 80386-based PC AT-compatible personal computers including those manufactured by Compaq, Wyse, Zenith, Olivetti, ITT and Texas Instruments. It also runs on IBM PC AT-compatible machines which have been 'turbocharged' with a 386 accelerator card such as the Intel INboard and Orchid Jet 386.

Dyalog APL/386 is a full implementation of Dyalog APL, but one that is specially configured for personal computers; and it includes support for standard PC keyboards, monitors and printers.

Not only is it a full second-generation APL with nested arrays, but Dyalog APL/386 supports very large objects and workspaces, up to a theoretical limit of 4 gigabytes!

Dyalog APL/386 uses 32-bit native 80386 code, and therefore takes full advantage of the 386 processor. Dyadic believes that as a result, Dyalog APL/386 outperforms all other APLs currently available for the PC. The following benchmarks were run on a 16MHz Compaq 386. Performance on the 20 MHz machine is even better.

Dyalog APL/386

'STSC' benchmarks: Compaq 386 with 80387 coprocessor 16MHz

(all figures are cpu seconds)

Small Ints	Large Ints	Floats	Mixed Fns	Scalar Ops	Fn Exec	Files
1.8	2.0	5.5	5.6	3.8	4.8	0.7

Dyalog APL/386 runs under SCO XENIX 386 System V Operating System which is available direct from Dyadic. In addition to supporting Dyalog APL, XENIX 386 provides a full 32-bit multi-user, multitasking environment; and unlike DOS, XENIX 386 releases the full potential of the Intel 80386 processor.

Dyalog APL/386 includes an interface to SCO CGI, a powerful device independent graphics development system, which is packaged with Xenix. It also includes a workspace containing business graphics functions for producing graphs, histograms, pie charts, titles and legends. The system supports standard PC displays, dot matrix printers, and other popular graphics devices.

Dyalog APL/386 Release 2 will include APL*PLUS/PC compatible full-screen functions designed to facilitate the transfer of APL*PLUS/PC applications to Dyalog APL.

Dyalog APL also supports SCO VP/ix, a system extension that permits multiple DOS and XENIX applications to be run concurrently from the system console. Character mode DOS applications can also be run from attached ASCII terminals.

This means that all the benefits of speed, size and functionality of Dyalog APL/386 can be obtained without sacrificing the ability to run DOS applications.

Dyalog APL/386 is priced at £995. SCO Xenix 386 Operating System costs £695.

Dyadic is pleased to welcome Janet Burton who has recently joined the Dyalog APL development team as a senior programmer.

MicroAPL Ltd

After five years at its Nine Elms premises, MicroAPL has decided to move. With effect from December 1 1987, we will be based at the South Bank Technopark. Here we will be close to nearly forty hi-tech companies, many of them developing exciting software and hardware products, in a building specially conceived by the South Bank Polytechnic to help leading-edge UK companies like MicroAPL gain from the synergy of close relations with each other and with the Polytechnic. The new address appears elsewhere in VECTOR, but it is:

MicroAPL Ltd.,
South Bank Technopark,
90 London Rd.,
LONDON SE1 6LN

...our new telephone number is (01) 922 8866 and the telex remains unchanged as 896885 IOTA.

On the product front, our entry level APL.68000 products are selling extremely well, especially in Europe. The recent price reductions to 99.95 (including VAT) for the Apple Macintosh, Commodore Amiga and Atari ST products has encouraged the introduction of a large number of new users to the delights of APL. The fact that a complete APL system (Atari ST and APL.68000) can be bought for a price which is similar to the price of the APL interpreter on other micros has not gone unnoticed in the APL community! Our free run-time policy on these small machines has stimulated the interest of APL software vendors in producing packages for these products.

The PC community has reacted enthusiastically to the APL.68000 implementation for the PC - MultiAPL. This hardware and software combination offers a particularly easy means of moving VSAPL applications onto the PC and the large amounts of RAM available (up to 4MB) free applications from the usual memory constraints on the PC.

Our users of Mirage-based machines may be interested to know that the move from the old factory has brought to our attention large amounts of second-hand stock and ex-demonstration equipment. When we have refurbished this equipment we will be circulating details. In the meantime if you think you might need APL terminals, printers (or even cheap cables!), why not give us a call?

More Mirage-based software is being released to try and tempt users away from APL. The latest language to appear is FORTH, which means that users now have a choice of nine programming languages to try. The Omnis database package will prove a powerful tool for many routine applications.

APL Impetus Ltd

Although APL Impetus Ltd was formed only in July this year, when it was decided that 'Impetus' no longer fitted into Boeing's product strategy, the product has been used by customers for 18 months and the concept goes back to 1976 when TABAPL, its mainframe timeshareing predecessor, was first marketed.

When the Impetus development was started in 1984, the challenge was to provide all the functionality of the mainframe product in a machine with only 640K memory. The successful result is based on APL*PLUS/PC, enhanced by 30 or so assembler routines for such crucial activities as formatting, scrolling and panning, and displaying without snow on colour and shaded monitors.

Demonstration diskettes are available for consultants and users who would like to know more about this alternative, APL-based, approach to solving those complex modelling problems which are beyond the scope of spreadsheets.

Mercia Software Ltd

APL*PLUS/PC Release 7 has just arrived and we are expecting a lot of interest from Release 6 users. Major new features include virtual workspaces, network sharing of APL files, monitoring function to keep track of execution time and a significantly improved graphics capability. Full details are available from Mercia.

APL development work has been concentrated on LOGOL, R.G. Brown's finished goods control system. Most of the effort has been going into writing graphics programs to illustrate the (often very sophisticated) statistical theory behind the system - turn lifeless figures into vital statistics as we say!

Cocking and Drury Ltd

Cocking & Drury have relocated their London office to 180 Tottenham Court Road. The move allows for further expansion and provides much improved public course facilities. Please refer to the Vendor Addresses list for the full address and telephone number. The Reading office address remains unchanged.

This year will see the launch of a new member of the APL*PLUS family of interpreters from STSC. The interpreter is designed for the new range of 80386 based personal computers, including the IBM PS/2 and the Compaq 386. Like APL*PLUS PC it will run under DOS, but will take full advantage of the 80386 architecture; 32-bit addressing will permit workspace and variable sizes that are limited only by the amount of installed memory. 32-bit integers and 1-bit booleans also feature.

In addition, the new interpreter will include the second-generation functionality of APL*PLUS UNX, APL*PLUS VMS and APL*PLUS Mainframe, e.g. nested arrays. High compatibility with APL*PLUS PC will offer a natural migration path at minimal cost for existing APL*PLUS PC users to the new generation of powerful personal computers

Release 7 of APL*PLUS PC and release 2.6 of STATGRAPHICS have been well received, including the release 7 Run Time System. Demand for updates has been much higher than expected, and we were particularly surprised at the number of people who have updated from release 5 to 7.

APL*PLUS PC Tools have been repackaged, with the 3270 IRMA comms now in a separate package; The IRMA Module. All the other utilities have been combined in a single toolkit.

The interest from VS APL users in the APL*PLUS Mainframe and Compiler technology continues, with several new trials planned for early 1988.

Cocking & Drury are supplying APL*PLUS VMS, the latest second-generation APL*PLUS interpreter for DEC MicroVAX and VAX computers running under DEC's native operating system. The product is treated in a similar manner to APL*PLUS Mainframe, i.e. full systems support as well as APL support are provided.

Cocking & Drury is developing a new service for companies concerned about the impact of APL on their computer services. Please contact Romilly Cocking for more details.

The past six months have seen four new consultants join the company; Liz Bland and Sylvia Kahl join the London office and Christian Jensen and Kevin Ryall are based in Reading.

Course Schedule

Statgraphics	10 Feb – 11 Feb	£280
APL Fundamentals	23 Feb – 25 Feb	£445
APL System Design	29 Feb – 3 Mar	£700
APL Fundamentals	8 Mar – 10 Mar	£445
Statgraphics	16 Mar – 17 Mar	£280
APL*PLUS PC Intermediate	21 Mar – 24 Mar	£525
APL Fundamentals	29 Mar – 31 Mar	£445
Statgraphics	6 Apr – 7 Apr	£280
APL Fundamentals	12 Apr – 14 Apr	£445
APL Fundamentals	26 Apr – 28 Apr	£445
APL System Design	9 May – 12 May	£700
APL Fundamentals	17 May – 19 May	£445

All prices exclude VAT. To book, or for further information, please call Beverley Satterley at:

COCKING & DRURY LTD.

THE APL PROFESSIONALS

180 Tottenham Court Road,
London W1P 9LE
Tel: 01 – 436 9481

Profit from our skills

Database

System
Design

System
Tuning

Project
Management

Statistics

O R

Accountancy

Forecasting

COCKING & DRURY LTD.

THE APL PROFESSIONALS

180 Tottenham Court Road,
London W1P 9LE
Tel: 01 - 436 9481

The (Nested) Education Vector

by Norman Thomson

Hurrah for a VECTOR dedicated to Education - how nice it would be to see Education rushing en masse to subscribe to VECTOR! Perhaps the best route to this goal is through statistics. In a note elsewhere in this edition, Peter Kelly from Newcastle University and I point out that there is no other high level language which provides a regression package free with every interpreter supplied!

In the closing address of the APL 84 conference in Helsinki, Fred Perkins of I. P. Sharp pointed out that computer users were increasingly package users, and as such have no reason for partiality or loyalty towards one underlying high level language rather than another. The success of APL therefore depended on the quality of the packages which were written and marketed by APL people; to carry on parading the merits of the APL language whilst ingoring this fact of marketing life was to hasten the doom of the very thing they were trying to promote.

There is however a cycle in the ways of computer usage in which the ever more sophisticated package user eventually becomes shackled by the restraints imposed by the software whose convenience gave him such delight not so long before. At this point there is no answer but to retreat again to the base language. In no subject area is this more true than in statistics, and it can be fairly said that APL must be the most flexible statistics package around.

Right at the start of statistical computing it is there with facilities for calculating simple descriptive statistics with ease equal to that of any of the popular packages. As noted above this even extends to multivariate regression, and if convenience of data-entry is taken into account, APL must surely come out as the top package at this level.

As statistical specialism increases it is undeniable that the speciality languages such as GLIM provide powerful algorithms on tap which are a rich and convenient source of ready-made number crunching, whose appeal outweighs the time required to provide the equivalent programs in APL. On the other hand the range of worthwhile statistical techniques is increasing at an exponential rate, and the special packages such as those mentioned can only contain a finite number of them, and that pre-determined at the time of last release. This is where the frustrated statistical user must surely be weaned back to APL. APL is there at the beginning and there at the end, the alpha and omega for the serious statistician with a computer.

So where does the education begin? Surely at the earliest introduction to statistics, be that at school or university. The case for the mathematics and statistics department having I-APL as a matter of routine is enormously strong, and we hope to see micro Labs up and down the country in which I-APL is as commonly available as spreadsheets and wordprocessors. By the time you read this I-APL will be available for PCs, and for BBCs not long afterwards. The benefit which can accrue from having individual members implant the idea of APL in the minds of their teacher friends is potentially enormous - personal recommendation is much the best advertisement. And if the response is "If APL is so good, why has't my Professor/LEA Adviser/Computer Science Department told me about it?" be quick to say "It's only now that you can afford it, so why settle for less than the best!"

Finally, and announcement which will be of great interest to APL enthusiasts in Universities and Polytechnics. An APL2 service is now available to them . . . for FREE! British academics who wish to try APL2 at no expense to themselves can now do so by applying to Warwick University. Withs IBM's support, Warwick have now got the ability to make their internal APL2 service available over JANET. Please announce this facility to any academic friends who may have been wanting to try APL2.

Detailed information can be obtained from:

Mike Hunt
Director of Computing Services
University of Warwick
Coventry CV4 7AL
Tel: 0203 523355

APL BOOKLIST

(In author order)

Title	Price £
Sharp APL Reference Manual, P Berry	10.50
Star Map, P Berry & J Thorstetsen	6.00
APL and Insight, P Berry and G Bartoli	4.50
APL 86 Tutorials, A Camacho (members £9.30)	12.00
A Source Book in APL, A Falkoff & K Iverson	10.00
FinnAPL Idiom Library	11.20
Application Systems in APL, Gibson Levine Metzger	30.00
APL:An Interactive Approach, Gilman & Rose	26.50
Solutions to Algebra, J Iverson	3.00
A Dictionary of APL, K Iverson	2.50
A Concise Dictionary of APL, K Iverson	2.00
Algebra: an Algorithmic Treatment, K Iverson	22.50
APL in Exposition, K Iverson	3.00
Applied Mathematics for Programmers, K Iverson	8.00
Elementary Analysis, K Iverson	8.00
Introduction to APL for Scientists & Engineers, K Iverson	3.00
Introducing APL to Teachers, K Iverson	3.00
Mathematics and Programming, K Iverson	8.00
APL Toolkit (CIPS APL SIG), R Levine	4.50
Reliable Software Through Composite Design, Myers	15.00
APL:An Introduction, H Peelle	POA
APL in Practice, Rose/STSC	40.00
Sharp APL Users Meeting Procs 1982 Vol 2	8.50
APL:Design Handbook for Commercial Systems, A. Smith	13.10
Resistive Circuit Theory, R Spence	20.00
Whizzbangs Volume II, R Sykes	14.50
Whizzbangs Volume I, R Sykes	14.50
An APL Notebook, Barrie Wetherill (when available)	1.90
APL Idiom List (Yale University)	2.00
APL 86 Conference Procs, D Ziemann (members £13.30)	17.30
APL Business Technology '83 Proceedings	11.20
APL Quote-quad the Early Years	32.00
APL Trivia Cards (per set)	4.50
Special APL 86 Wallet Offer (see Vector 3.2 page 92)	22.50

Please order direct from

WARWICK UNIVERSITY BOOKSHOP

Arts Centre, University of Warwick, Coventry CV4 7AL, Tel 0203-523388

Access, American Express and Barclaycard accepted. Order by telephone.

Postage at cost on credit card orders.

APL Product Guide

Compiled by Cathy Dargue

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:

Cathy Dargue,
60 Downhall Ley,
BUNTINGFORD,
Herts SG9 9TL.
Tel: 0707-325161 ext 2418

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	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	20,000+	Multi-user APL computer using 68020 CPU. Std. configuration 2Mb RAM, 16 RS232 ports, 68 Mb hard disc, 720K diskette
	SPECTRUM	7,000+	Expandable multi-user APL computer using Motorola 68000. Std. conguration 1 Mb RAM, 12/36 Mb disc, 12 ports.
	Atari 1040ST	799	1 Mb Mono/Colour System, includes 1 Mb disc drive & mains transformer built into Console

APL TIMESHARING SERVICES

COMPANY	PRODUCT	PRICES(£)	DETAILS
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 7	475	STSC's full featured APL for IBM PC, PC/AT and PS/2. Now with virtual workspace, quad-MF, GSS graphics, Network and VGA support.
	Upgrade 6 to 7	145	
	Upgrade 5 to 7	260	Extension upgrades to release 7.
	Run-time	poa	Closed version of APL*PLUS/PC which prevents user exposure to APL.
	APL*PLUS UNX	poa	STSC's 2nd generation APL for IBM PC/AT, DEC, AT&T and other Unix computers.
	APL*PLUS VMS	poa	Integrated 2nd generation APL for DEC m/cs under VMS
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	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.
	APL*PLUS/MAC	poa	APL for the Macintosh. Big workspaces, big objects, mouse, icons etc
MetaTechnics	APL*PLUS Rel 6	475	Discount on quantity
MicroAPL	APL.68000	1,000	a Full implementation with component les, error trapping etc. for SPECTRUM, HP300, SUN, NCR etc.
	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 Apple Macintosh	257	
	APL.68000 Commodore Amiga	200	Full APL interpreters with support for windows, mouse, graphics etc.
	APL.68000 for Atari ST	170	
	APL.68000 for IBM-PC	995	
	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 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 -PL terminal, 50-19200 baud, 15f8if5 anti-reex screen, low prole keyboard
	Tandberg TDV 2271	1,195	Combined APL/ANSI ergonomic terminal as above.
Gen. Software	Mellordata	400	Second-hand Elite 3045A
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
	Concept201	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	APL-385	50(PC),125(mf)	including ...
	FSM-385		Screen development
	DRAW-385		Screen design
	DB-385		Relational W.S.
	GEN-385		Miscellaneous 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.
	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.
APL IMPETUS	Impetus	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)	SHAREFILE & enh'ments	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	poa	Full featured FMT for APL2
	WSDOC	5,000	
	FILEMANAGER	8,000	
(microcomputer)	STATGRAPHICS Rel 2.6	645	Integrated Statistics and graphics on IBM PC, AT, PS/2 and compatibles
	Update 2 to 2.6	140	
	Update 1 to 2.6	330	
	APL*PLUS PC Tools	325	Utilities including: RAM disk, full screen data entry, menu input, report generation, file documentor, exception handling and games.
	IRMA Module	120	327x IRMA support.
	Fin & Stat. Library	350	Financial & statistical routines
	SPREADSHEET MGR	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 - consisting of: PC & mainframe modules
	RMS		Relational databases.
	AMS		Multi-dimensional arrays
	RAMS		Combined RMS & AMS.
	BMS		Dynamic Financial modelling & forecasting
	FMS		Full-screen handler for APL*PLUS/PC. (AP124-based)
	CMS		Communications package
	SOS	poa	Scheduled ordering and stock control.
FASTCODE	FASTCODE, MONITOR	$239	Assembler written monitor for APL applications in APL*PLUS/PC
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 21	585	Integrated stat. graphic system for PCs.
	Upgrade to Release 21	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
	APL Spreadsheet	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	240	APL*PLUS/PC spreadsheet-like data entry and validation system.
	The APL DEBUGGER	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 Consultants	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.
Buckland Management Systems	Consultancy	poa	Business and Technical systems in commerce and industry. Designing, programming and implementing applications of O.R and statistics.
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	Junior consultant
		140-200	Consultant
		185-300	Senior consultant
		275-400	Managing consultant
Peter Cyriax	Consultancy	100-150	Junior Consultant
		120-200	Consultant
		160-300	Senior 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.
Uniware	Consultancy	call	Junior to managing consultancy in APL.

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 Tel: 617-246-0300
APL 385	Adrian Smith	Brook House, Gilling East, York. Tel: 04393-385
APL Consulting	Jill Moss	17 Barton Street, Bath, Avon BA1 1HQ Tel: 0225-62602
APL Impetus Ltd	Cedric Heddle	Rusper, Sandy Lane, Ivy Hatch, SEVENOAKS, Kent TN15 0PD Tel: 0732-885126
APL People	Valerie Lusmore	17 Barton Street, Bath, Avon. Tel: 0225-62602
APL Software Ltd	Philip Goacher	27 Downs Way, Epsom, Surrey KT18 5LU Tel: 03727-21282
		17 Barton Street, Bath, Avon BA1 1HQ Tel: 0225-62602
APL Software Tech ology	John Hagger	14 Rosewood Avenue, Alveston, Bristol BS12 2PP Tel: 0454415737
	Per Hultin	46 Vicarage Road, South Benfleet, Essex SS7 1PB Tel: 0374550501
Beta-plan APS	Kim Andreasen	Stengrade 75 , DK-3000 Helsingor, Denmark. Tel:45 2 21 48 48
Buckland Management Systems	John Buckland	Westwood, 9 Grove Road, Camberley, Surrey, GU15 2DN Tel: 0276 684327
Butel Technology Ltd	Mike Munro	Butel House, 3 Great West Rd., London W4 5QJ Tel: 01-995-1433
Anthony Camacho		2 Blenheim Road, St. Albans, Herts AL1 4NR. Tel: St. Albans (0727) 60130
Paul Chapman		18, Trevelyan Road, London, SW17 9LN Tel: 01-767 4254
Cocking & Drury Ltd.	Romilly Cocking	180 Tottenham Court Road, LONDON, W1P 9LE16 Tel: 01436 9481
		155 Friar Street Reading RG1 1HE. Tel: 0734-588835
Datatrade Ltd.	Tony Checkseld	38 Billing Road, Northampton, NN1 5DQ. Tel: 0604-22289
Delphi Consultation	David Crossley	Church Green House, Stanford-in-the-Vale, Oxon SN7 8LQ. Tel: 03677-384
Dyadic Systems Ltd	Peter Donnelly	Park House, The High Street, Alton, Hampshire Tel: 0420-87024
E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS. Tel: 0689-24741
Farnell International Instruments Ltd.	R. Fairbairn	Jubilee House, Sandbeck Way, Wetherby, W. Yorks. Tel: 0937-61961
	Roger Attard	Davenport House, Bowers Way, Harpenden, Herts. Tel: 05827-69071
FASTCODE	Andrew Dickey	P.O. Box 281, Croton-on-Hudson, New York 10520, U.S.A. Tel:(914) 271-3200
General Software Ltd.	M.E. Martin	22 Russell Road, Northolt, Middx. UB5 4QS Tel: 01-864 9537

H.M.W. Programming Consultants Ltd.	Ken Jackson	142 Feltham Hill Rd, Ashford, Middx. TW15 1HN. Tel: 07842-41232
IBM UK Ltd	Chris Sell	PO Box 32, Alencon Link, Basingstoke, Hants. RG21 1EJ Tel. 0256-56144
Inner Product Ltd.	Dominic Murphy	Eagle House, 73 Clapham Common Southside, London SW4 9DG. Tel: 01-673 3354
Lloyd Savage Ltd	Philip Johnson	Cambridge House, Oxford Road, Uxbridge Middx, UB8 2UD Tel: 0895-59826
Mercia Software Ltd.	Gareth Brentnall	Aston Science Park, Love Lane, Birmingham B7 4BJ. Tel: 021-359 5096
MetaTechnics Systems		John Stenbridge Unit 216, 62 Tritton Road, London, SE21 8DE. Tel: 01-670 7959
MicroAPL Ltd.	Bernadette Leverton	South Bank Technopark, 90 London Road, LONDON SE1 6LN Tel: 01-922 8866
Modern Business Technology Ltd. (MBT)	Michael Branson	P.O. Box 87, Guildford, Surrey GU4 8BB Tel: 0486823956
M.T.I.	Ray Cannon	7 Pine Wood, Sunbury-on-Thames, Middx. TW16 6SH Tel: 09327 80848
Parallax Systems Inc	Kevin Weaver	60 West 9th Street, New York, New York 10011, U.S.A. Tel: 212-475-4001
Peter Cyriax Systems	Peter Cyriax	213 Goldhurst Terrace, London NW6 3ER Tel: 01-624 7013 (Answerphone) 0860-377963 (Mobile)
Portable Software	Richard Smith	60 Aberdeen Ave, Cambridge, Mass. U.S.A. 02138. Tel: 617-547-2918
QB On-Line Systems	Philip Bulmer	5 Surrey House,Portsmouth Rd Camberley, Surrey, GU15 1LB Tel: 0276-20789
The Rochester Group	Robert Pullman	164 Pinnacle Rd., Rochester NY 14620 Tel:716-461-3169
Shandell Systems Ltd.	Maurice Shanahan	12 High Street, Chalfont St. Giles, Bucks HP8 4QA. Tel: 02407-2027
I.P.S.A. Ltd.	David Weatherby	10 Dean Farrar Street, London SW1. Tel: 01-222 7033
Sugar Mill Software Corp.	Lawrence H. Nitz	1180 Kika Place, Kailua, Hawaii 96734 Tel: (808) 261-7536
Tektronix UK Ltd.	Paul Morgan	Fourth Avenue, Globe Park, Marlow, Bucks SL7 1YD. Tel: 06284-6000
Uniware	Eric Lescasse	15 Rue Erlanger,75016 Paris, France Tel: (1) 45-27-20-61 Telex: 648348F UNIWARE

Book Review

An Introduction to APL for the IBM PC & XT by William H. Murray and Chris H. Pappas,

Paperback, 167pp text plus 50pp appendices index etc
Published by Brady Communications Co,
Price £21:70.

Reviewed by Anthony Camacho

I believe this is a very bad book. In fairness I should add that not everyone will agree with my opinion and if you don't I hope you will be able to tell from what I write whether you would like it better than I do. The rest of this review is devoted to a description of the book and justification of my opinion.

The authors try to cover two popular APLs for the PC XT (& AT); STSC's APL*PLUS/PC and IBM's APL/PC. There is no mention of Sharp APL or PortaAPL.

They begin by relegating the set up of the machine to the appendix. The fourth paragraph of the introduction reads: "If you do not desire to install the keyboard labels, or if you plan to move your APL from one computer to another, you will need the keyboard overlay provided with this book. Carefully remove the overlay, prepare it according to the directions and place it over your keyboard." - there are no directions; the "overlay" is a cardboard sheet with a sort of perspective drawing of the IBM keyboard on one side and the STSC keyboard on the other; the keys are not marked consistently and some of the markings are wrong. Iota is marked as a mirror image "S"!

The chapters are:

1 - Instant APL
2 - Simple APL programming
3 - Edit, print & file
4 - Introduction to function writing and workspaces
5 - System functions, variables and special features
6 - Workspaces and libraries
7 - Making music
8 - Creative graphics in IBM & STSC APL
9 - Towards more advanced APL programming

A - Installing the APL system
B - Optional diskette package
C - Graphics with STSC's PocketAPL
D - Using the Microsoft mouse with APL
Index

As you can see from this list, chapters 1-6 are about the basics of APL in STSC's and IBM's implementations and chapters 7 & 8 are a bit of time off to play with sound and graphics. The chapter on sound tries also to be a music primer and includes MARYSLAMB and FURELISE. The chapter on graphics puts the words "Graphics is Great" in a box and draws a sine wave (misprinted by rotating it 30 degrees) and a synthesised square wave. Chapter 9 has a subtitle "Additional examples" and contains examples of encode, decode, matrix division and the equals outer product. It also contains a very elementary telephone directory function of over a hundred lines.

Now you know what's in the book I will deal with some niggles and then with my major objections.

Niggles

1. Typesetting

The book is wholly set by a professional typesetter who doesn't understand what shapes the APL symbols are. He matches them as well as he can from his stock of standard symbols, so the quad is square, the circle is much too big, log is printed as circle overstruck with multiply and there is no consistency in the size of jot in lamp, execute, format or on its own (when it is sometimes printed as a degree). In short the typesetting is bad.

2. Proofreading

In several places the instructions to the typesetters have been set and left in the text (eg"do not wrap function lines"). In many places a matrix result is split between two pages. On page 75 a line of BASIC which was intended to set Y to the cube of I is typeset as I and 3. Rounding is explained as the floor of 0.555 + (I do hope this is a proofreading error!). On page 5 the text has 18 percent when the function has .015. A vital branch arrow is omitted on page 49. A good many of these and other errors make nonsense of the text. Perhaps a learner would not be able to puzzle them out.

3. Incomplete explanations

Many explanations are ambiguous or incomplete or fail to make the point. Their example of high minus 359 minus high minus 57 would give the same result with two subtractions. The paragraphs on monadic domino only cover square matrices and in my view fail to explain them. Incidentally the only conditional branch I could find in the examples was of the form 'branch to label times iota result of test': there is no explanation that label is assigned the value of its line count on opening the function and that the iota ensures that (in index origin one) multiplication will be by one or the empty vector; nor do they explain the effect of index origin zero on this construct. Nowhere could I find the essential part of the explanation: that a branch to the empty vector is parsed as 'branch to nowhere; ie don't branch'.

Serious Weaknesses

The two serious weaknesses are that the authors do not use words carefully enough and that the book sets a bad example.

First the words:

"You will soon discover that many of the APL symbols have both a dyadic and monadic use of operation."

"Since our right argument will have two dimensions our left argument must also have two dimensions."

(This is about take and drop! It continues: "You will use a two element vector.") Notice the slide from "our left argument" to "You will use". The point of view from which the book is written changes unpredictably. Some function keys "Move you to . . " some "Allow you to . . " some "Enable you to . . ".

"F6 Allows you to copy a line. First you move the cursor to the line you want copied and press F&. Be patient. Many of the F(n) commands take several seconds to perform their operations. After several seconds, if you look very carefully at the bottom of your screen, you will see that the 6:COP command prompt has changed to 6:COP*. This is to indicate that the copy option has been turned on. Next move the cursor to the line <u>after</u> the line that is to be copied and press F6 a second time. Again after several seconds the, 6:COP*, command prompt will return to its original state, 6:COP, and finally the line will be copied."

(sic Explanation of F6 in IBM editing). I think they mean "<u>after</u> the place where you want the line copied to . . "

"<Ctrl-A> Allows you to alter the structure of the object you are defining, or editing. In our example above, the bottom line of the screen should now look like this: Editing: FIBONACCI[V]6 The V inside [] indicates that you are defining a vector. Press <Ctrl-A>. You will notice that at the bottom of the screen there is now a prompt asking you to define the definition of this object. Now press the letter F for function. Immediately you return to the original screen and if you notice, the bottom of the screen has changed to: Editing: FIBONACCI[F]6 <Ctrl-A> allows you to define a (F)unction, (M)atrix, or a (V)ector."

(sic From the explanation of STSC editing).

Second the examples:

Here is part of the introduction to the function TELEPHONE (IBM version 105 lines; STSC version 101 lines):

"True believers in APL would probably have preferred that we broke this large function down into several smaller, independent functions. We could have done that - and would have done that - except that we felt some clarity might have been lost in having to look at numerous functions."

One wonders how they come to think that a 100 line function is clearer than ten 10 line functions. These functions are just like BASIC with branches to labels (like GETIT or TOHERE) at the head of subroutines.

Here is a final example - the function NUMBERGUESS from halfway through the book.

```
[0]  NUMBERGUESS
[1]   ⎕IO←1
[2]   MYSCORE←0
[3]   COMPSCORE←0
[4]  AGAIN:'I WILL PICK A NUMBER FROM 0 TO 10   YOU HAVE THREE
     GUESSES.'
[5]   NUMBER←(?11)-1
[6]   'ENTER YOUR FIRST GUESS'
[7]   MYGUESS←⎕
[8]   →HWIN×ι(NUMBER=MYGUESS)
[9]   'ENTER YOUR SECOND GUESS'
[10]  MYGUESS←⎕
[11]  →MWIN×ι(NUMBER=MYGUESS)
[12]  'ENTER YOUR THIRD GUESS'
[13]  MYGUESS←⎕
[14]  →LWIN×ι(NUMBER=MYGUESS)
[15]  COMPSCORE←COMPSCORE+10
[16]  →SORRY
[17] HWIN:MYSCORE←MYSCORE+10
[18]  →REPORT
[19] MWIN:MYSCORE←MYSCORE+7
[20]  COMPSCORE←COMPSCORE+3
[21]  →REPORT
[22] LWIN:MYSCORE←MYSCORE+3
[23]  COMPSCORE←COMPSCORE+7
[24] REPORT:'YOU GOT IT (',(⍕NUMBER),') -- THE SCORE IS:'
[25]  →SCORE
[26] SORRY:'YOU DIDN''T GUESS IT, IT WAS ',⍕NUMBER
[27] SCORE:'YOU: ',(⍕MYSCORE),'     COMPUTER : ',(⍕COMPSCORE)
[28]  ' '
[29]  ' '
[30]  →AGAIN
```

In short, apart from the niggles, what is wrong with this book is that it sees APL purely as a programming language similar to BASIC with a few minor differences such as array variables, yet it fails to give any idea of APL's power and range in that role.

If your view is that APL is a commercial programming language and you bought a commercial interpreter for your PC then you will want to use it for a better class of work than you get here: this half-baked introduction is no way to begin down that road.

If you agree with me that APL should be seen primarily as a notation (in my view that leads to better programs too) then the failure of this book to provide the fundamental groundwork or to cover the behaviour of functions with unexpected (eg empty) arguments, disqualifies it. Paul Berry and Gilman and Rose still rule - OK?

AN INITIAL LOOK AT I-APL

by Dave Weatherby

Introduction

This Christmas I got the opportunity to try out a pre-production version of I-APL, the shareware APL that was born at APL86 in Manchester. I-APL's objective is to enable anyone, especially schools, to try out a good quality implementation of APL on the machines that they already possess. I-APL is designed to be run on almost any micro with a minimum of 64K memory, and versions should soon be available to run on the BBC and RML micros which are the most popular in British schools.

In my view the long term future of APL will depend on its acceptance in schools and so I jumped at the chance of a preview of I-APL. However a family Christmas and reviewing APL do not fit very comfortably together and therefore this article should be seen as an initial look at I-APL rather than an in depth review. In any case I have only been able to use I-APL on the AMSTRAD PC512, an IBM PC clone, whereas a full review needs to test out its performance on a number of different machines, especially the BBC micro. In addition the supplied workspaces that I was given were not the final versions.

What You Get

The complete package costs £11 plus postage and packing and consists of the following:

Floppy Disc with APL interpreter and Workspaces
An Instruction Manual (52pp)
Tutorial (48pp)
Language Encyclopaedia (296pp)

The floppy disc may be copied and distributed freely, although the manuals are subject to normal copyright. So if you already know APL or prefer to use it in conjunction with other APL books then I-APL can be yours for the price of a floppy. The I-APL committee have decided not to distribute the floppy on its own but they will supply the manuals individually. I imagine that the various Shareware distribution groups will be happy to provide copies of the floppy at their normal rates.

Getting Started

The supplied floppy disc had a READ.ME file that explained the parameters I needed to specify to get APL characters on my CGA screen. Hercules and EGA are also supported and for those without any graphics an option is provided which uses ASCII symbols instead. I-APL loaded within 10 seconds - a real relief from the 40 seconds or so that I am used to with STSC APL. Typing ⎕AV displayed upper and lower case, normal and underlined characters as well as all the standard APL characters plus a few extras such as match, epsilon-underlined and the British pound sign. Resolution on my CGA screen was reasonable, although the overprinted characters are not easy to read.

Keyboard Support

APL's special character set is a big advantage in communicating easily with the computer, but poses problems for APL implementers and popularisers. The I-APL ASCII option mentioned above does not provide a keyword APL as pioneered by MicroAPL; it merely gives a one-to-one arbitrary mapping between APL and ASCII symbols. The authors of I-APL clearly expect users to have access to APL characters. All the documentation uses APL symbols exclusively and would therefore require those with no APL symbols to perform a continuous translation. Despite the attempt to make the ASCII symbols relate to their APL equivalents I believe that very few people are likely to get through this process.

I-APL allows upper case characters to be entered from the keyboard using the shift key and the APL characters are entered in normal lower case. Upper and lower case ASCII characters can be entered by toggling the INSERT key. I-APL follows IPSA and STSC in providing a keyboard mapping of APL characters that is a union with ASCII. For example the minus and plus signs are in the same position as for the standard ASCII keyboard, rather than in that of the traditional APL keyboard mapping. The STSC and IPSA union keyboards are slightly different and I-APL has a totally different mapping again, so that there are now at least 3 different union keyboards. It might be better named the disunion keyboard!

I-APL has chosen to link the APL character keyboard position to that of the similar ASCII symbol. So epsilon is lower case E and iota is lower case I. However more controversial is the decision to link the not equals symbol to lower case Z due to similarity of shape, or grad up and grad down to lower case q and lower case d respectively. The overall effect is that the keyboard mapping is quite different from any other APL I have seen. While this may not be a problem for newcomers to APL it will cause them irritation if they graduate to other more powerful APLs.

I do not believe that the mnemonic nature of I-APL's layout is sufficient to justify a wholly new keyboard layout, especially as those mnemonics rely on lower case letters while most keyboards are labelled with upper case characters. I suggest that I-APL follows the union keyboard approach of normal ASCII for the first and shift character sets with more specialised APL characters available via the INSERT toggle key.

The APL

The APL itself conforms to the draft ISO APL standard and even offers some facilities beyond the standard. For example grad up and grad down work on character matrices, and the replicate extension of compress as well as stop and trace are all provided. No file access or screen driver facilities are provided. However a method of calling machine code is included, which I understand will be used to create workspaces offering these and other facilities.

During my use of I-APL I did not notice any bugs and was certainly never dumped back in DOS, despite hitting keys at random while the interpreter was executing. Hitting CTL-C did confuse the screen display somewhat, however my general impression was that the APL is solid.

A good implementation of direct definition is also provided and, although its use in large scale applications may be controversial, it undoubtedly simplifies function creation as well as encouraging a modular programming style. Functions may also be defined using the the familiar del function. Direct definition functions are treated slightly differently from standard functions in that they cannot be suspended and appear in listings with a colon as the last character of their name. An example of the care which has been taken with this APL is that ⎕FX has been extended to create direct definition type functions. A function is also provided to convert from direct definition form to del form.

I hope very much that other APL vendors follow I-APL's lead in offering direct definition as an integral part of their APLs.

Variable types include 1-bit boolean, 2-byte integers and 6-byte reals. I-APL has therefore sacrificed some accuracy to minimise memory requirements. On my Amstrad I had a useful workspace size of 31624 bytes and, since I-APL only makes use of 64k of memory, I assume that the workspace size would be very similar on other machines. The documentation states that compromises between speed and memory have always been resolved in favour of memory in order to maximise the workspace available.

User Interface

The user interface is simple. In-line editing is provided, but there is no scrolling memory. The last line executed can be recalled for editing but not any previous lines. However the interface is intelligent enough to define the last line as the last immediate execution line and ignores system and editing commands.

I am very used to scrolling memory and would have willingly swapped workspace size for a page or two of scrolling session log. I suspect that beginners will miss it even more. It would be very attractive to have a parameter to allow multiple page session logs at the expense of workspace size.

Speed

Given that speed has been sacrificed for space it would be surprising if I-APL was very fast. The table of benchmarks shows a comparison with STSC APL and clearly indicates, as expected, that STSC APL is probably at least 15 to 20 times faster in normal circumstances:

BENCH MARK FOR I-APL AND STSC APL VERS 6.3
ALL TIMES IN SECONDS ON AN AMSTRAD PC512

OPERATION		I-APL	STSC
Plus reduction	z←+/vi	1.49	0.43
Logical reduction	z←∨/vl	0.14	0.005
Maximum reduction	z←⌈/[1]mi	0.44	0.023
Exponentiation	z←vi*.1	95.9	26.220
Absolute value	z←|vr	0.68	0.38
Indexing	z←vr[vi[⍳20]]	0.48	0.017
Sorting	z←vi[⍋vi]	5.15	0.45
Take	z←¯2 1↑mr	0.15	0.022
Membership	z←vi∊vi	3.44	1.81
Transposition	z←2 1⍉mc	1.66	0.056
Outer product, characters	z←vc∘.=vc	2.22	0.072
Outer product, integers	z←(⍳50)∘.+⍳50	7.76	0.208
Inner product, reals	z←vr⌊.+vr	2.63	1.37
Matrix division	z←mr⌹10↑vr	7.03	0.36
Fibonacci series (note 2)	z←fib	28.78	2.21
Multiplication	z←vr×3.14	2.87	1.36
Division	z←vr÷3.14	7.95	1.82
Logarithm	z←⍟vr	39.08	13.98
Sine	z←1○vr×.1	39.47	17.95

Note 1: The variables used in the benchmarks are defined by

```
mi←10 10⍴vi←(500⍴0 1 0 0 1)/⍳500
vl←1 0 1 1 0 0 0 1
mr←10 10⍴vr←vi+0.1
mc←26 26⍴vc←'abcdefghijklmnopqrstuvwxyz'
```

Note 2: The function fib is defined as

```
∇r←fib
  r←1 1
  1:→(100>⍴z←z,+/¯2↑z)/1
∇
```

The more important point however is whether I-APL is fast enough for teaching purposes. In my view the results show that I-APL is certainly fast enough for educational purposes and may even be fast enough for real applications in some circumstances.

Documentation

I was given a pre-release version of the documentation and cannot comment on its presentation, and so the following remarks are restricted to content only.

1. Tutorial. The Tutorial by Linda Alvord and Norman Thompson is clearly designed for those with a theoretical bent. The terms 'scalar' and 'vector' are introduced quickly and are followed up with the use of the term 'argument' with very little explanation. However I assume that the expected audience will be largely teachers of maths and science and so the presentation is probably appropriate. The topics covered emphasise probability theory, matrix inversion and the use of simple graphics to investigate functions. Again the emphasis is on the use of APL in the teaching environment. Given the assumed target audience of teachers and theoreticians the Tutorial provides a good introduction to APL. It is unlikely however to appeal to the teenage hacker or computer freak.

2. Instruction Manual. The Instruction Manual is designed to show how I-APL runs on a particular machine. I was very impressed with the level of detail provided about the way data is stored and how the APL primitives are calculated. Such information may not be used often but gives real confidence in the integrity of the system. It also emphasised the rigour of APL in comparison to languages such as LOGO. My main criticism of the Instruction Manual is that the instructions on how to load APL were relegated to an Appendix. It would be preferable to have a section labelled 'Getting Started' for those users, such as myself, who cannot wait to read all the documentation.

3. Encyclopaedia. The Encyclopaedia attempts to give information about the I-APL language in general rather than any particular implementation. I did not get enough time to really assess the quality of the information given but what I saw appeared well written and comprehensive. However I was not convinced that the format was well designed for easy reference. For example system functions such as Lnc are to be found in the list of contents under 'Name lists'. This is not helpful when trying to understand someone else's code. Moreover any item given in the list of contents is not given in the index which requires the user to look in two places. The encylopaedia needs a better index and one that attempts to index APL characters in a similar manner to that in Paul Berry's Sharp APL.

Overall the quality of the documentation was good, although it would have been useful to have had a brief introduction to what was on the floppy and what each of the manuals were for. The orientation of the documentation is geared to the theoretic or academic user of computers rather than the home or small business user wanting to get the computer to do something useful. Personally I believe this orientation is correct but it is important that the package gets exposure to the right audience.

I think it would be very helpful to include a practical application of APL, such as the workspace NATIONS as presented at the Helsinki APL conference by Paul Berry and Michael Berry. Such a workspace illustrates how simply APL can create database type applications as well as providing good examples of APL code. Although the importance of reading APL as a means of learning has been stressed by Iverson, the I-APL documentation provides little opportunity for doing so.

Supplied Workspaces

I am unable to say much about the supplied workspaces since those that I saw were not the final versions. However I am told that workspaces will be available which provide the APL functions in Linda Alvord's book on probability, in the APL Tutorial and in the APL Encyclopaedia. In addition workspaces will be provided that give access to simple turtle colour graphics, give access to DOS files, provide functions similar to `⎕WPUT` and `⎕WGET` in STSC APL and offer simple APL tutorials. Clearly some of these workspaces are specific to the PC, but, no doubt, similar facilities could be provided on machines such as the BBC.

Overall Assessment

I-APL for the PC is a full implementation of the APL standard with some workspaces that offer significant extensions. The orientation of the product is for teachers of analytic subjects such as maths and probability. The availability of a colour graphics workspace should enable I-APL to emulate LOGO's turtle graphics and this should help APL to be accepted by the education community. Overall this is an impressive product and deserves to be a Shareware classic.

Availability

I-APL (ISO APL with graphics, file access and screen handling)
I-APL Ltd., 2 Blenheim Rd., St Albans

PRICE: Software free, documentation £11

APL*PLUS/PC Version 7.0

reviewed by Adrian Smith

Background

APL*PLUS/PC is now firmly established as the world standard in APL interpreters for the IBM PC and compatibles. Clearly a new release is of great importance to the APL community, as it immediately supersedes all previous versions and thus moves a substantial part of world APL development into a new environment.

The purpose of this review is to help you to decide where it might pay you to upgrade from previous versions, where you should stay put, and possibly where you might be better off staying with a previous release. As you will see, APL*PLUS/PC 7.0 has many useful new features, but there are also some drawbacks which you may like to consider before upgrading.

Features and Benefits

Some of the new things you can do in version 7.0 are:

- networking. If you are into LANs (as I regret to say I am not yet) you can use ⎕FSTIE, ⎕FSTAC, ⎕FRDAC, and ⎕FHOLD to do the expected things to preserve file integrity across the network.

- set up a disk file to let APL swop objects automatically in and out of memory. A fair amount of detailed control is possible over the rules by which objects are swopped. This lets you move 'big' applications straight down from 2Mbyte mainframe systems and run them without modification.

- monitoring execution time with ⎕MF. To some extent the fact that you have a dedicated processor in the PC makes this an unnecessary luxury, but I agree that it is very convenient to be able to use your mainframe functions (assuming you are running APL*PLUS mainframe) to run monitoring checks on the PC.

- direct graphics output to a wide variety of devices which support the CGI standard. If you use you APL with one of these 'virtual device interfaces' you can run a smaller version of the interpreter without all the old ⎕G-style graphics functions.

- use the PS/2 to its full capability with 'mode-8' graphics and a downloaded APL font. The VGA graphics in particular I have found fast and impressive; the resolution is now the same as the old Hercules mono standard, but of course you have colour as well.

- VT100 emulation is an option in terminal mode. Your PC can thus be used as a simple monochrome terminal into a VAX system, or into a mainframe with an asynch protocol converter.

In addition there are some new assembler functions, more printers are supported, and there are some speedups to the primitives and to the workspace housekeeping.

Drawbacks and Reservations

So much for the good points. Essentially these echo the STSC press release, and you can find out more details from any of the vendors. Some of the things they don't tell you are:

- it is a lot bigger (160K against 124K for release 5). You therefore lose some 40K of workspace, and it takes rather longer to load up (16sec from floppy). Even the 'space-saving' version only gets you 10K or so back. What's more you still can't (legally) get rid of the copyright banner. I'm afraid I still tend to implement on 4.2 for this reason alone. (see 'Hackers Corner' for the right places to prod it if you do want your own sign-on banner).

- it is about 20% slower across the board than release-5. There is no measurable difference between 6 and 7, other than in the published speed-ups. As soon as you allow you arrays to stray over the 32K limit, all the timings roughly double. See the table of 'Smith Benchmarks' for full details.

- earlier releases run quite happily on the VGA adaptor using the EGA characters and graphics modes. Personally I strongly dislike the APL font supplied for the VGA, and have replaced it with one of the APL fonts I did for my Hercules card. I shall get a copy of this onto the software exchange as soon as I can. Of course you can use VGACHAR.COM from your Release 7 disks with any earlier release of APL*PLUS/PC if that is what you want.

- the VT100 emulation looks extremely flakey! I have failed to find any decent documentation (other than the location to ⎕POKE!) on its use; it also has all sorts of peculiar side-effects (undocumented as far as I can tell) such as resetting ⎕WINDOW and the default background colour. When running as a VAX terminal it was fine in simple terminal mode, but rapidly became totally confused in a form-based application (e.g. SQL*CALC); when running as a mainframe terminal through a protocol converter it just made a total dog's breakfast of the screen. I think this one should go back to the drawing board; it is a good idea which deserves proper implementation!

In many ways, I feel that this release of APL*PLUS/PC is a step backwards. One of the great benefits of the PC is the absolutely consistent response time; as soon as you bring in the concept of paging, you are back in the 'mainframe' environment where the same operation could take differing lengths of time depending on how you got there. To a lesser extent the extension to objects over 32K gives a similar set of problems; a trivial change to the user's data could suddenly double execution times.

From my own experience, it is generally far better to fail the application on `WS FULL` or `LIMIT ERROR` and to fix the code to use less space. A typical example was a 'summarize this by that' expression which did it the VS APL way using plus-dot-times and jot-dot-equals; on Vn5 this failed on `LIMIT ERROR`, so we moved it to Vn6. It then started to fail on `WS FULL`, so we took 10 minutes to rewrite it (extract unique elements and loop round them with plus-dot-equals). It is now back on Vn5, and runs some 30 times quicker than it did before! I have a horrible suspicion that with Vn7 we would just have given it the workspace and hence given the user a slower and less predictable system.

The same sort of thing applies to a little workspace I use to process the output from this word-processor so that I can get it up to our mainframe for laser printing. A typical document is around 20K, so Vn5 handles this quite happily. Moving to 33K forces you to Vn6, and suddenly it is like watching paint dry. I hate to contemplate the possibility of running a complete VECTOR (approx 120K) through Vn7; much better to invest the time in a rewrite which moves down the file in (say) 10K slabs.

In summary, the IBM PC is a 5 year-old architecture, based around 64K segments and a 1Mbyte address limit. Trying to make it pretend that it is a mainframe with 32bit addressing and infinite memory is like teaching a dog to walk on its hind legs; it can be done, but it makes terribly inefficient use of the resources available. Use your PC as a PC, and I think you will be most pleasantly surprised at the power and performance available; use it a an ersatz mainframe and I fear that you can only expect disappointment.

Benchmark Results

Here again is my standard collection from VECTOR 4.1. As previously noted, the simple 1000 looper FN 1000 seems to be the best indicator of overall speed:

```
APL Expression                 Type          Vn. 5  Vn. 7
--------------                 ----          -----  -----
T←⎕TS ⋄ +/V+V   ⋄ DO ⎕TS-T ⍝ INT ....        350    380
T←⎕TS ⋄ +/V[I] ⋄ DO ⎕TS-T ⍝ INT ....          50     60
----------------------------------------------------------
T←⎕TS ⋄ V+.×V   ⋄ DO ⎕TS-T ⍝ RL  ....        380    440
T←⎕TS ⋄ +/V*2   ⋄ DO ⎕TS-T ⍝ RL  ....       3680   3850
T←⎕TS ⋄ X←+/M   ⋄ DO ⎕TS-T ⍝ RL  ....        660    710
T←⎕TS ⋄ V+.×,M ⋄ DO ⎕TS-T ⍝ RL  ....        2140   2200
----------------------------------------------------------
T←⎕TS ⋄ +/S='T' ⋄ DO ⎕TS-T⍝ CH  ....        3630   3790
T←⎕TS ⋄ 'FAT'∊S ⋄ DO ⎕TS-T⍝ CH  ....          60     60
T←⎕TS ⋄ X←S∊'FAT' ⋄DO ⎕TS-T ⍝ CH ...        2520   4730
----------------------------------------------------------
T←⎕TS ⋄ X←B∨B^~B ⋄ DO ⎕TS-T ⍝ BL....        4840   5550
T←⎕TS ⋄ ^/B       ⋄ DO ⎕TS-T ⍝ BL....        110    160
T←⎕TS ⋄ X←∨\B     ⋄ DO ⎕TS-T ⍝ BL....        220    280
----------------------------------------------------------
T←⎕TS ⋄ FN 10     ⋄ DO ⎕TS-T ⍝ FN....        330    440
T←⎕TS ⋄ FN 100    ⋄ DO ⎕TS-T ⍝ FN....       3100   4010
T←⎕TS ⋄ FN 1000   ⋄ DO ⎕TS-T ⍝ FN....      30.3s  41.7s
----------------------------------------------------------
T←⎕TS ⋄ X←FF^.='EJD..'   ⋄ DO ⎕TS-T  ⍝    60     60
T←⎕TS ⋄ X←FF^.=⍉20 5⍴'EJD..'⋄DO ⎕TS-T⍝  33.6s  37.5s
==========================================================

All timings on IBM PC with NEC V20 processor (Approx
20% faster than standard 8088-based PC).
```

Conclusion

If you need to run on a PS/2 and want to use VGA graphics, you need version 7. If you must (and I mean must) handle large objects in large workspaces, then you need release 7. If you want really detailed analysis of execeution times, or need networking support, or could use the VT100 emulation, then you need release 7.

If none of the above apply, then stick with release 5. It is smaller, faster, and probably better debugged. Personally, I shall continue to stockpile both 6 and 7 in my cupboard, and install from my release 5 disks (or even occasionally from 4.2) unless there is a proven need for the new facilities.

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 attend. It also covers other selected events which are likely to be of interest to the wider APL community.

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

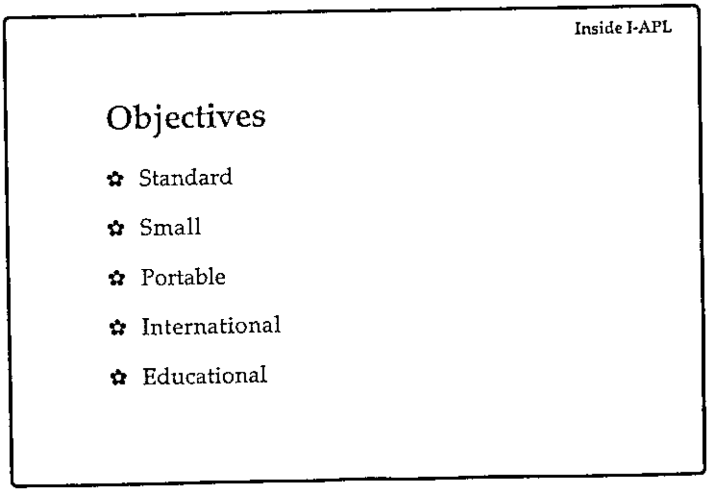
The Activities Officer (see inside back cover) will respond enthusiastically to offers from individuals who wish to contribute seminars and supporting papers.

I-APL: Under the Bonnet

Talk by Paul Chapman, notes by Eileen Wilks

I-APL is an APL interpreter specifically designed to run on small machines which are suitable for schools. Paul Chapman wrote I-APL and gave a very interesting talk to the British APL Association about its development. He had limited finances to work with, and the work was completed in about 5 months

Paul began by describing his objectives:

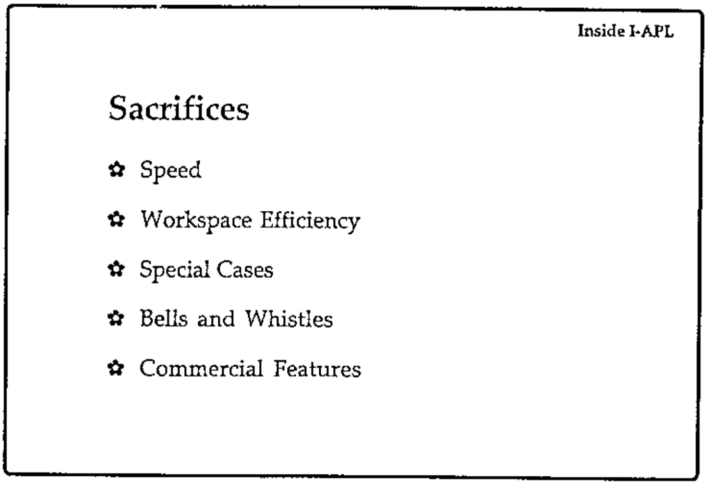


Inside I-APL

Objectives

- ✿ Standard
- ✿ Small
- ✿ Portable
- ✿ International
- ✿ Educational

Because of the tight restrictions imposed on him, certain sacrifices had to be made:

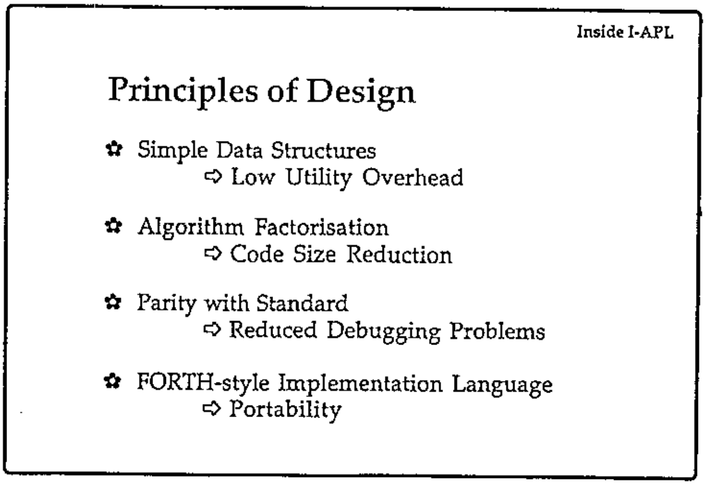


Inside I-APL

Sacrifices

- ✿ Speed
- ✿ Workspace Efficiency
- ✿ Special Cases
- ✿ Bells and Whistles
- ✿ Commercial Features

Paul then went on to describe the design policy he adopted.

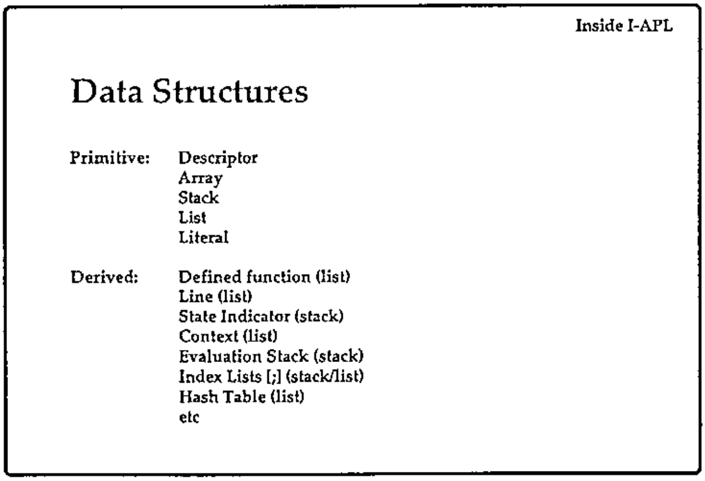


Inside I-APL

Principles of Design

- ✿ Simple Data Structures
  ⇨ Low Utility Overhead
- ✿ Algorithm Factorisation
  ⇨ Code Size Reduction
- ✿ Parity with Standard
  ⇨ Reduced Debugging Problems
- ✿ FORTH-style Implementation Language
  ⇨ Portability

He then outlined the data structures required ....



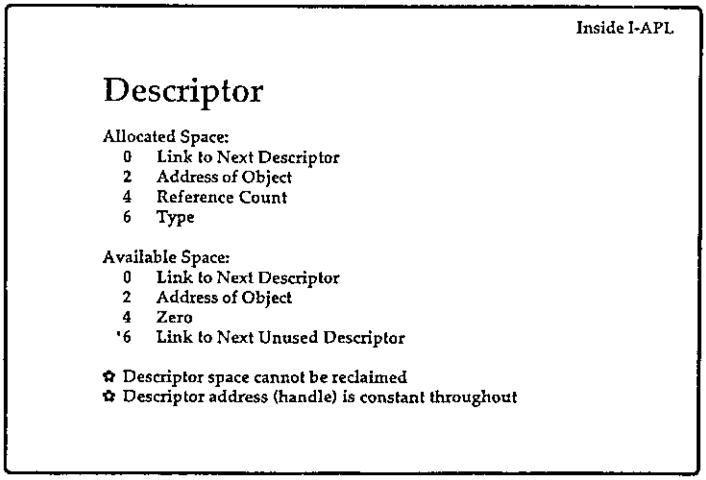
Inside I-APL

Data Structures

Primitive: Descriptor
Array
Stack
List
Literal

Derived: Defined function (list)
Line (list)
State Indicator (stack)
Context (list)
Evaluation Stack (stack)
Index Lists [;] (stack/list)
Hash Table (list)
etc

.... and went on to describe them in more detail.



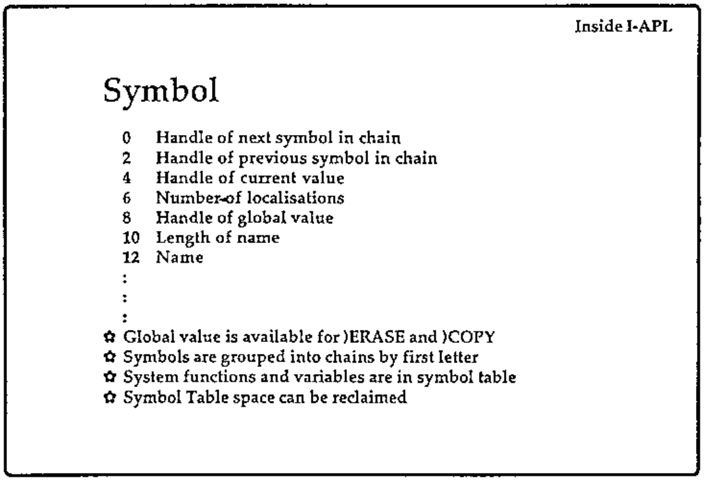
Inside I-APL

Descriptor

Allocated Space:
0 Link to Next Descriptor
2 Address of Object
4 Reference Count
6 Type

Available Space:
0 Link to Next Descriptor
2 Address of Object
4 Zero
'6 Link to Next Unused Descriptor

✿ Descriptor space cannot be reclaimed
✿ Descriptor address (handle) is constant throughout

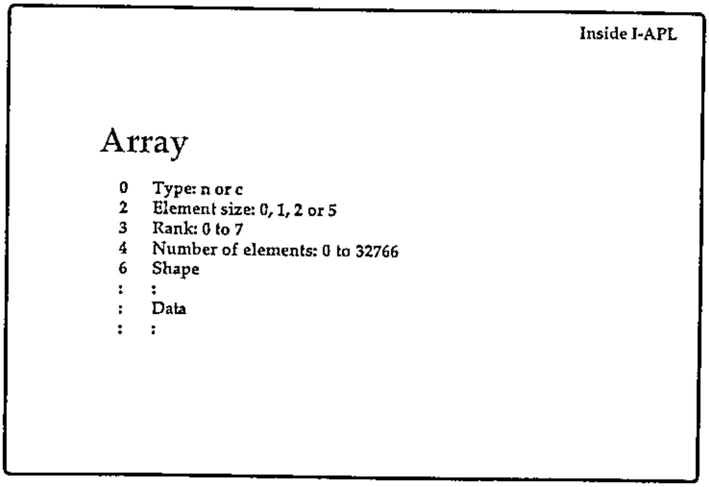


Inside I-APL

Symbol

0	Handle of next symbol in chain
2	Handle of previous symbol in chain
4	Handle of current value
6	Number of localisations
8	Handle of global value
10	Length of name
12	Name
:	
:	
:	

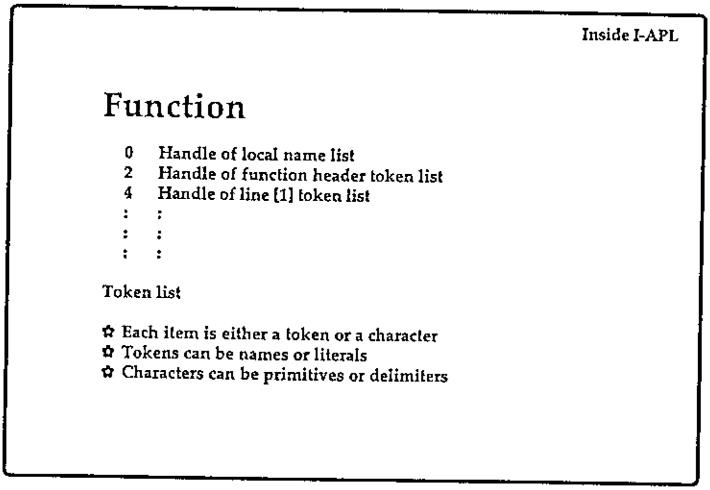
✿ Global value is available for)ERASE and)COPY
✿ Symbols are grouped into chains by first letter
✿ System functions and variables are in symbol table
✿ Symbol Table space can be reclaimed



Inside I-APL

Array

0	Type: n or c
2	Element size: 0, 1, 2 or 5
3	Rank: 0 to 7
4	Number of elements: 0 to 32766
6	Shape
:	:
:	Data
:	:



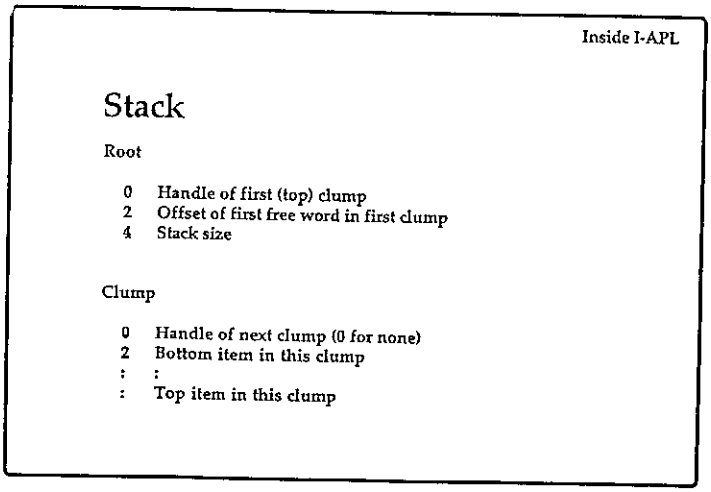
Inside I-APL

Function

0 Handle of local name list
2 Handle of function header token list
4 Handle of line [1] token list
: :
: :
: :

Token list

✿ Each item is either a token or a character
✿ Tokens can be names or literals
✿ Characters can be primitives or delimiters



Inside I-APL

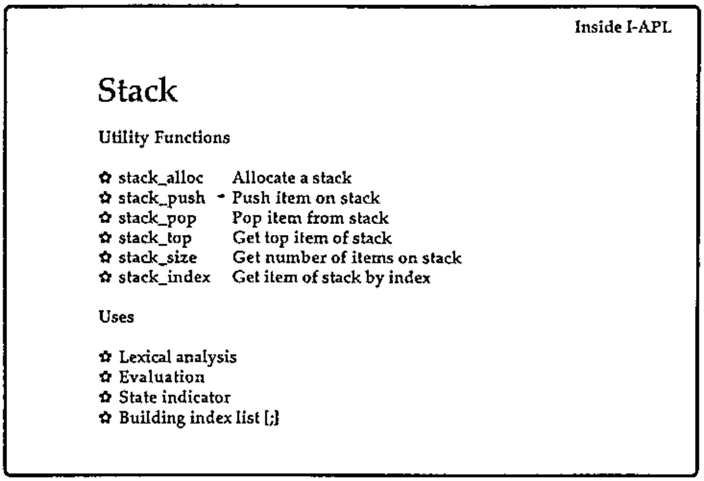
Stack

Root

0 Handle of first (top) clump
2 Offset of first free word in first clump
4 Stack size

Clump

0 Handle of next clump (0 for none)
2 Bottom item in this clump
: :
: Top item in this clump


Inside I-APL

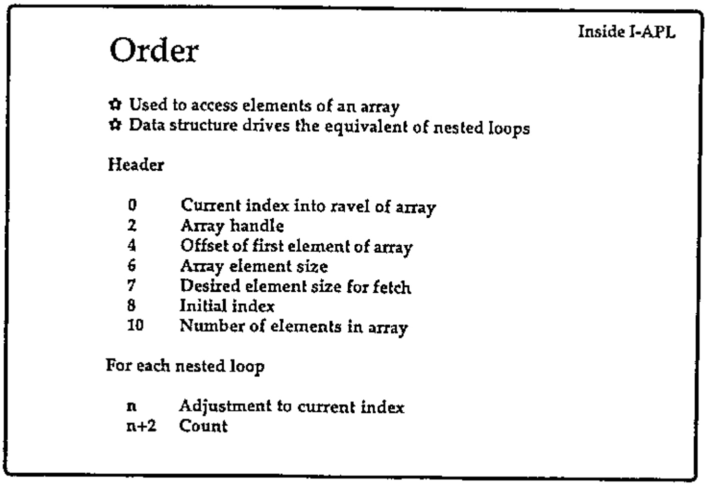
Stack

Utility Functions

✿ stack_alloc	Allocate a stack
✿ stack_push	Push item on stack
✿ stack_pop	Pop item from stack
✿ stack_top	Get top item of stack
✿ stack_size	Get number of items on stack
✿ stack_index	Get item of stack by index

Uses

✿ Lexical analysis
✿ Evaluation
✿ State indicator
✿ Building index list [;]


Inside I-APL

Order

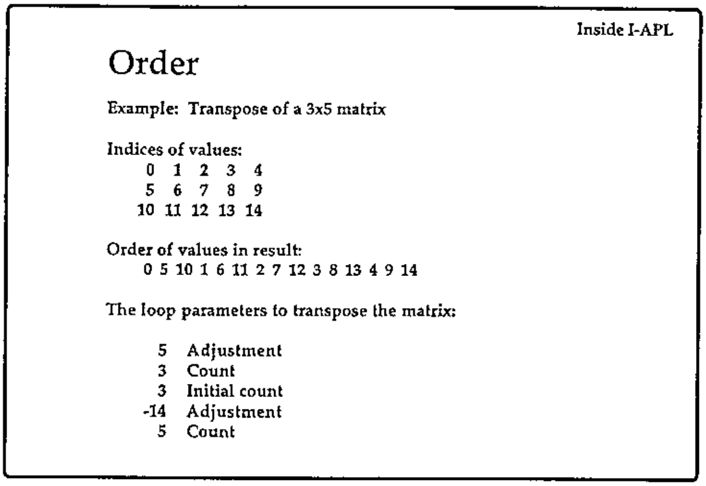
✿ Used to access elements of an array
✿ Data structure drives the equivalent of nested loops

Header

0	Current index into ravel of array
2	Array handle
4	Offset of first element of array
6	Array element size
7	Desired element size for fetch
8	Initial index
10	Number of elements in array

For each nested loop

n	Adjustment to current index
n+2	Count



Inside I-APL

Order

Example: Transpose of a 3x5 matrix

Indices of values:

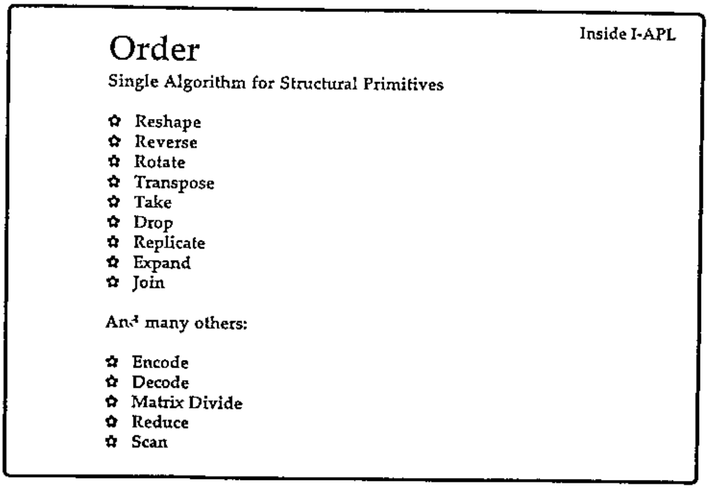
```
 0  1  2  3  4
 5  6  7  8  9
10 11 12 13 14
```

Order of values in result:

```
0 5 10 1 6 11 2 7 12 3 8 13 4 9 14
```

The loop parameters to transpose the matrix:

```
  5  Adjustment
  3  Count
  3  Initial count
-14  Adjustment
  5  Count
```



Inside I-APL

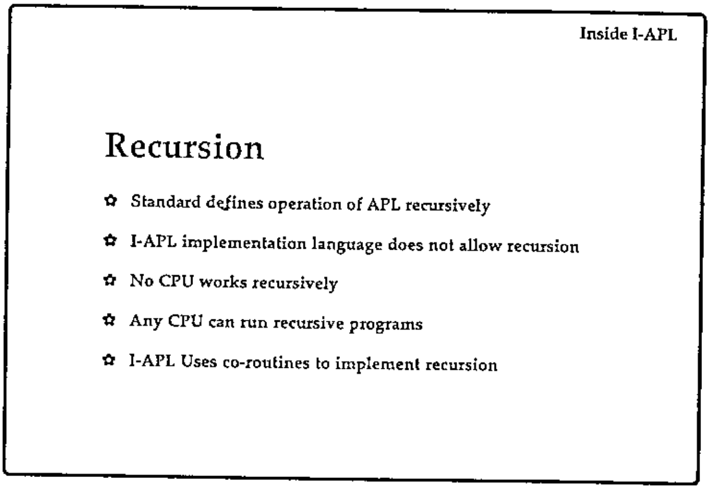
Order

Single Algorithm for Structural Primitives

✿ Reshape
✿ Reverse
✿ Rotate
✿ Transpose
✿ Take
✿ Drop
✿ Replicate
✿ Expand
✿ Join

And many others:

✿ Encode
✿ Decode
✿ Matrix Divide
✿ Reduce
✿ Scan

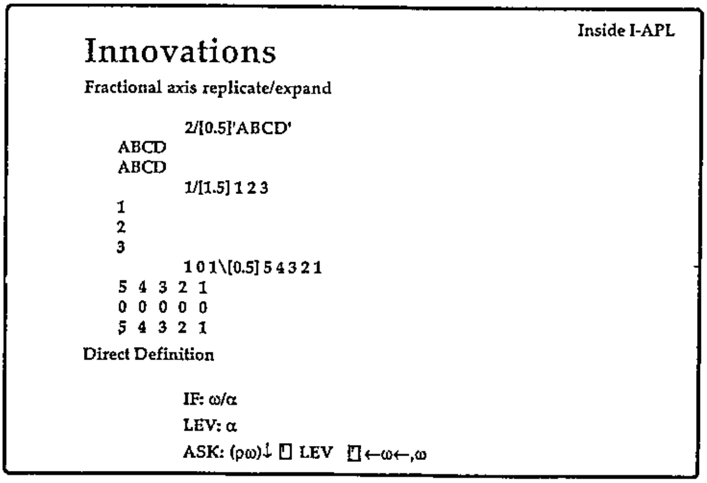


Inside I-APL

Recursion

- Standard defines operation of APL recursively
- I-APL implementation language does not allow recursion
- No CPU works recursively
- Any CPU can run recursive programs
- I-APL Uses co-routines to implement recursion

He concluded the talk by pointing out some new ideas which had been incorporated into I-APL.



Inside I-APL

Innovations

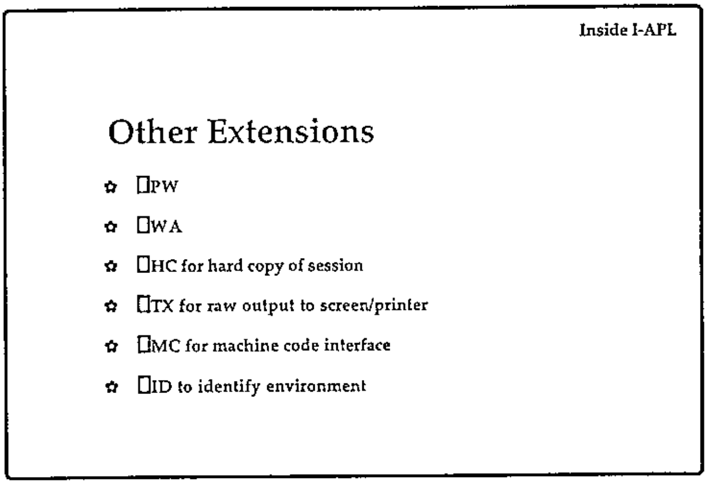
Fractional axis replicate/expand

```
          2/[0.5]'ABCD'
    ABCD
    ABCD
          1/[1.5] 1 2 3
    1
    2
    3
          1 0 1\[0.5] 5 4 3 2 1
    5 4 3 2 1
    0 0 0 0 0
    5 4 3 2 1
```

Direct Definition

```
          IF: ω/α
          LEV: α
          ASK: (ρω)↓ ⎕ LEV  ⍞←ω←,ω
```

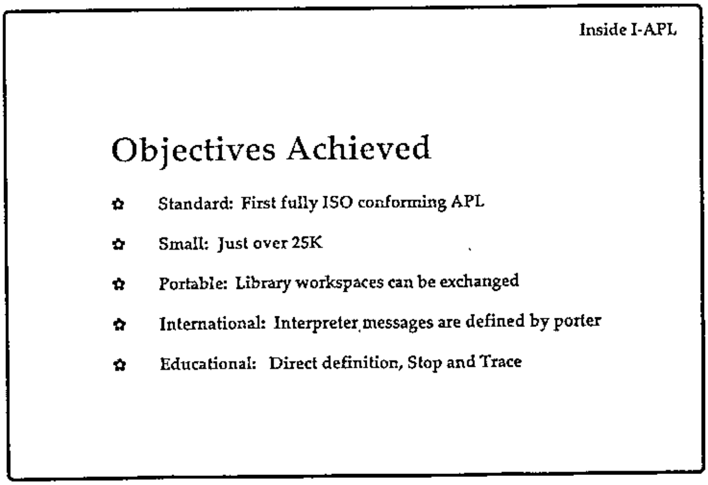
... and some extra features.


Inside I-APL

Other Extensions

- ⎕PW
- ⎕WA
- ⎕HC for hard copy of session
- ⎕TX for raw output to screen/printer
- ⎕MC for machine code interface
- ⎕ID to identify environment

Finally, he looked back at his initial objectives to show how they had been acheived.


Inside I-APL

Objectives Achieved

- Standard: First fully ISO conforming APL
- Small: Just over 25K
- Portable: Library workspaces can be exchanged
- International: Interpreter messages are defined by porter
- Educational: Direct definition, Stop and Trace

At the end of his talk, a member of the audience stood up to say that writing an ISO-standard APL interpreter in less than 25K was brilliant. Paul Chapman agreed!

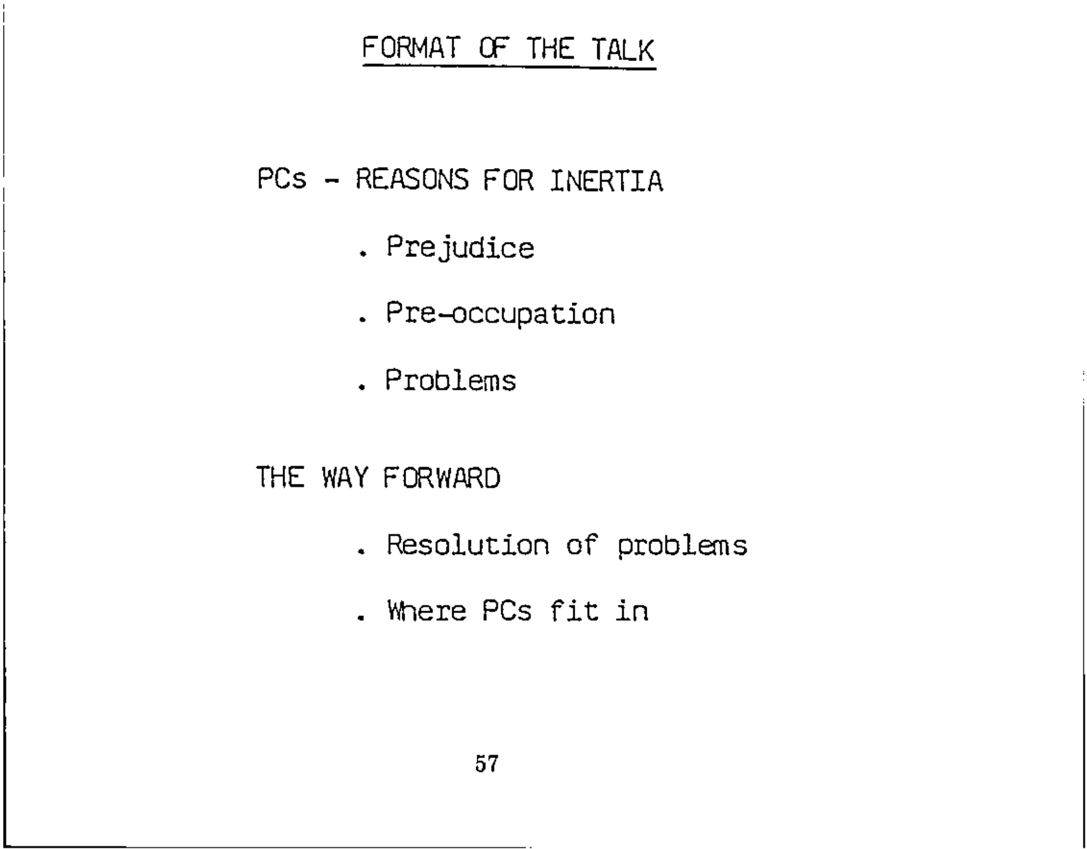
From Mainframes to Small Machines

Talk by Paul Smith, notes by Adrian Smith

Introduction

Paul is OR Project Manager with British Gas plc (South Eastern), formerly British Gas South Eastern, formerly SEGAS, originally the South Eastern Gas board! He told us all this at the beginning, partly to see who still thinks of them as 'the Gas Board', but mainly to make the point that the ground under any system keeps changing. Company names are only part of the problem.

Thanks anyway to Paul for passing on copies of his foils; the annotations are a mix of his own marginal notes, and my own recollection of his talk.



FORMAT OF THE TALK

PCs - REASONS FOR INERTIA

. Prejudice

. Pre-occupation

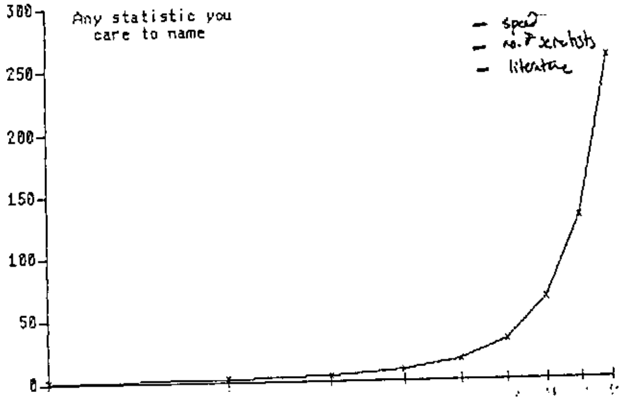
. Problems

THE WAY FORWARD

. Resolution of problems

. Where PCs fit in

Firstly, why the prejudice? The accelerating rate of change of almost anything can be illustrated by:



... and the resulting generation gap is making it harder and harder to stay close to the 'leading edge'.

```
                    TECHNO-GENERATION GAP


            Definition

            The smallest age difference
            between people who don't
            understand each other


            Size

            5 years and dropping
```

In summary, Paul has one very good reason for inertia when it comes to moving away from mainframes to smaller machines: prejudice.

A Short History of APL in BG(SE)

APL started its career with 'a real wow' back in 1980 when work was begun on 2 major systems:

APL in BGSE - POTTED HISTORY

1979	Acquired VSAPL, APL*PLUS
1980	Work starts on 2 major systems
1982	1st versions of both systems go live
1984	2nd Version of one system goes live
1986	Work starts on 3rd system

In between all this there has been lots of hassle (not so terrific!) and much use of APL for statistics, general analytical work and suchlike small one-offs (much more effective). Lots of lessons have been learned ...

LESSONS WE HAVE LEARNED

. Large systems require special skills

. Large systems can require a lot of maintenance

. A 'house-keeping' system to store, organise and document functions is very useful

. A small number of full-screen utilities can go a long way

... but APL is still far from being the 'wow' it was in 1980. The kind of things that Paul finds are wrong with APL are illustrated by the following (hypothetical??) dialogues:

```
A:  (after a lot of muttering)
    ...(expletive deleted)

B:  What's up ?

A:  It's taking ages to write
    this program - full-screen
    stuff. It's horrible.

B:  What are you trying to do ?

A:  Oh, it's ...

B:  There's a utility to do
    that, you know ?!

A:  Well, it's not in my list !

B:  I expect it's out of date

A:  Hasn't been updated, more
    likely - typical!
```

The worst case on record involved the reinvention of dyadic iota! <<I should talk ... I once wrote a very nice 'membership' function using or-reduce and jot-dot-equal ... Ed.>>

Some other examples of the intrinsic hostility of APL when it comes to maintaining large systems ...

ANOTHER TYPICAL SCENE

A: Is there any way to ... ?

B: Yes, I had a function to do that once - saved a lot of time

A: Can I have a copy of it please ?

B: Ah, well, I'm not sure which workspace it's in - it was some time ago

(after several unsuccessful attempts to)COPY)
I know, ask Tony, I got it from him in the first place

A: He's on holiday

B: Oh yes, sorry.

... or to finding your way round someone else's workspace:

```
)LOAD BODGEMARCUS
SAVED 15:37:58 10/06/87
DESCRIBE
VALUE ERROR
      DESCRIBE
            ^
```

```
       )FNS
ABOVE    ADD      ADDDUPS  ADDNULLS          ADDOUTPUT         ALERT    ALFASORT
ANCESTORS         APLTEST  ASK      ASKMENU  ASKNAME  ASKNNI   ASKNUM   ASKSTOP  BESIDE
BINCHARS∆TO∆NUMBERS        BLANKDUPS         BLANKPREV         BLANKZERO         BMSUP
BODGE    BREAK    BREAKERROR        BROTHERS          BUILDDATA         CENTRE
CHANGEGLOBALS     CHANGELOC         CHANGELOCFORM     CHANGEMOD         CHANGEREP
CHECKCPU          CHECKFILES        CHOOSESYSTEM      CHOOSESYSTEMOLD   CLEARERRORS
CLEARSCREEN       CLEARUSERS        CLOSEOUTPUT       CMS      CMSREAD  CMSSTACK
COMMAND  CONFIRM  CONVERT  CONVERTDT         COPY     COUNTLINES        CPRINT
CREATEFILE        CSPRINT  CSREAD   CSREAD1  CTOP     DATALINES         DATE
DEFINE∆EXPORT     DELAYCLEAR        DENY     DESCENDANTS       DMTB     DOAS     DOAUTH
DOBATCH  DODATA   DODEFS   DODT     DOEXPORT          DOFCS    DOGLOBALS         DOINFO
DOLS     DOMARCUSLINK      DONAMES  DONEWLS  DONIAS   DONIASLINK        DONIASLIST
DOOLDLS  DOPHIN   DOREDUNDANT       DOREPQF  DOREPORT          DORUN    DOSREPS  DOSUB
DRAWTREE          DROPDUPROWS       DROPDUPS          DTB      DTCHARS  EDERR    EDITALL
EDITALLCMD        EDITLOCS          EDITMESSAGE       EDITPF00          EDITPF01
EDITPF02          EDITPF03          EDITPF04          EDITPF05          EDITPF06
EDITPF07          EDITPF10          EDITTEXT          EDMSG    ELDERBRO
ENSURETYPE        ENTERFILE         EPSREAD  EPSWRITE          ERASESCREEN       ERRIMM
ERROR    ERRORDETAILS      ERRORTRAP         EXPA     EXPANDDT          EXPB     FAPPEND
FATHER   FDATAAPPEND       FDATAGET          FDATAREAD         FDATAREPLACE      FGET
FIELDATTR         FIELDTYPE         FILESIZECHECK     FILESTATUS        FILETIME
FILLOUT  FINDTEXT          FINDTREE          FINREAD  FIVETOFOUR        FIXUPREPS
FIXUPSEL          FIXUPSELS         FNAME    FORMAT   FOURTOFIVE        FRAME1

FREEFORMAT        FREPLACE          FTIMEREP          FUDGE    FUDGENIAS         FCHK
GETDESTINATION    GETEXTSOURCES     GETFIELD          GETFIELDS         GETGLOBALNAMES
GETGROUP          GETINDICES        GETMEMOGLOBALS    GETOUTPUTFORM     GETSOURCES
GET∆EXPORT        GET∆LOCTOTALS     GLOBAL   GLU      GLU2     GLU3     GO       GROUP
HANDLER  HOUSE    IF       IMMWR    INDENT   INDENTDES         INDEX    INTEGERISE
INTENSITY         LASTNONZERO       LEVEL    LISTFNS  LJUST    LOCAL    LOCKOUT
LOOKUPSUM         LPRINT   LSLOC    MATRIX   MATRIXFI          MATSORT  MENUMAN  MIM
MSG      NAMEINUSE         NEWLOCS  NIASFMT  NUMBERS∆TO∆BINCHARS        NUMERIC  N∆
OBJECTINDEX       OFFHOLD  ONEANDONLY        OPENFILES         OPENOUTPUT
OVERNIGHT         PACK     PANDRED  PARTSUBANALYSIS   PMATRIXFI         PMATRIXVI
PMAXRED  PORRED   PORSCAN  PPLUSRED          PPLUSSCAN         PRINTCSDATA
PRINTLINES        PRINTSHEETS       PUTBACK  QAS      QASTYPE  QCALC    QCPU
QFORMCALC         QGROUP   QLS      QMODCALC          QTYPE    QTYPE1   READFORMAT
READGLOBALS       READLSDETAILS     READMODDATA       READOBJECT        READOBJECTINT
READSCREEN        READSUBJECT       REFORMAT          RELEASENAME       RJUST
ROWSINERROR       RUSS     RWTD     SAVECALCDETAILS   SAVEGLOBALS       SAVELSDETAILS
SEARCH   SELECT   SETCURSOR         SETPF    SETUPERROR        SETUPLS  SHARE    SHIFTL
SHIFTR   SHOW     SHOWTIME          SIZE     SONS     SQUEEZE  START    STARTCHANGE
STARTNEW          STARTUP  STARTUPOLD        STOP     SUBTOTAL          SUBTREE
SYSTEMMESSAGE     TAKELATEST        TAPEFILE          TESTCOPY          TESTGROUPS
TEXTTIME          TIME     TO       TRACEDT  TRANSFERLS        TRANSLATE         ULH
ULH1     ULINE    UNIQUE   UNIQUEROWS        UNLOCK   UNPACK   UPDATECALC
UPDATEDATA        UPDATELSGLOBALS   UPDATE∆EXPORT     UPD∆LOCTOTCALC    VALIDTREE
VERDUMMY          VERNOS   VERSEL   VIEW     WHO      WNG      WNGMSG   WNGSTORAGE
WRITE    WRITEMODDATA      WRITEOBJECT       YESORNO  ZAP      UPPERCASE
```

In the light of this experience, it is hardly surprising that PCs are approached with some caution, and plenty of the aforementioned prejudice.

To begin with

PCs ARE DIFFERENT

- Different editor
- Native files are nasty
- Full-screen utilities don't work
- Slow
- Size problems

Result An alien environment

... and the last thing Paul wants to do is to multiply his current problems:

CURRENT PROBLEMS

- Knowledge mostly confined to a few experienced people
- There is a wide range of abilities/experience
- Utilities are not readily accessible
- There are few standards for coding, documentation or specifications

... by including a new, untried environment in his APL systems.

Proposed Way Forward

Paul concluded his talk with a set of foils outlining his proposed path towards a machine-independent environment. He made it clear that it is still early days, and that the strategy could be forced to change if any elements of the proposed approach cannot be implemented in practice.

PROPOSED WAY FORWARD

A machine-independent APL development environment, featuring:

- Functions and panels on file
- Applications defined as grouped subsets of functions
- Utilities held on a single central file
- Ability to copy from other users' libraries
- Panels maintained using panel design utlities
- Data files held in standard format, with useful info

BENEFITS OF PROPOSED APPROACH

- Standardisation
- Applications structured
- Documentation
- Access to utilities (single version)
- Guidelines for new users
- No extra learning for PCs
- Portability
- Transferability

DISADVANTAGES

- Does not support 'flashy' features of PCs
- Incompatible with PC packages
- Has to be very robust

WHERE PCs FIT IN

INAPPROPRIATE for:

- Number-crunching
- Large volumes of mainframe data

APPROPRIATE when:

- The mainframe becomes unworkable
- Your end-user only has a PC
- You want to share applications with others
- You need interfaces to other PC-base software

Summary

Paul summarized his talk by emphasising three points:

- APL is a powerful language for developing systems - on mainframe or on the PC.
- To maximize its usefulness, full advantage must be taken of utilities and standard methods
- Avoiding learning different methods for different machines outweighs the benefits of 'nice to have' features peculiar to one machine.

It was interesting that the comments from the floor fell neatly into two camps. Firstly there were the 'old school' mainframers who found it hard to see why anyone should want to move to a PC in the first place; secondly there were the 'new wave' who had never used anything but APL*PLUS/PC, and failed to understand how anyone could possibly work on anything else.

My owm comment to Paul at the time was that Rowntrees had started the move to PCs about 2 years ago, with much the same stated aims. In particular we ensured that we had the same set of Full-screen and file utilities available on both sides of the divide so that we could move easily from one system to the other.

In reality the real 'instant winners' on the PC front have been systems which do take advantage of all those 'PC-ish' things like fast graphics and pop-ups; we have indeed moved some existing mainframe work down to the PC, but it is quite true that all you end up with is a slow, isolated 'mainframe' which gives the user little benefit other than free CPU time.

ORIGINS

by Graham Parkhouse

The question of origins in APL may be put simply as follows: "How should we index 'A' from the alphabet 'ABC . . Z'? To get 'A', should we ask for 'ABC'[0] or 'ABC'[1]?" In APL we can choose which way we must ask for it by means of the index origin, quad-IO, a "system variable" which may take the value 0 or 1.

```
      ⎕IO←1
      ALPH←'ABCDEFGHIJKLMONPQRSTUVWXYZ'
      ALPH[1 2 3]
ABC
      ⎕IO←0
      ALPH[1 2 3]
BCD
      ALPH[0 1 2]
ABC
```

APL has an inverse indexing function, iota, index of, which works like this:

```
      ⎕IO
0
      ALPH⍳'ABC'
0 1 2
      ⎕IO←1
      ALPH⍳'ABC'
1 2 3
```

Iota has another meaning when it has no left hand argument; it returns the first n integers, n being specified in the right hand argument.

```
      ⎕IO
1
      ⍳5
1 2 3 4 5
      ⎕IO←0
      ⍳5
0 1 2 3 4
```

From this example we might say it is convenient to consider quad-IO as holding the value of the "first integer". Is the first integer 1 or 0?

So we come to the point of forming an opinion of whether we prefer the index origin to be 1 or 0. At the November 1987 meeting of the British APL Association it was revealed by a show of hands that there was a preference in the audience for 1, though most present were prepared to use either 1 or 0 depending upon the circumstances. My experience is that computer scientists are zealous origin noughters.

The choice would probably not be given if one of the options resulted in obvious benefits; if that were so the choice would have been made for us. Iverson appears to be convinced that origin 0 is the right choice, and he makes it mandatory in Dictionary APL. There is no doubt that having a choice has many disadvantages simply because so many mistakes are made by incorrectly mixing code written for different index origins. But whether you choose 1 or 0 you will be able to do just the same computations. Some expressions turn out slightly more elegantly with one rather than the other, but such elegance is no more significant than the omission of an odd "-1" or so otherwise needed in an expression. A reason why origin nought might rightly be slightly favoured on the grounds of elegance is that "true" and "false" have been (arbitrarily) chosen as 1 and 0. Had they been chosen as 1 and 2 the situation would have been reversed.

Suppose when God created the universe he had made time discrete, and for all we know he may have done it like that. We tend to think of time as being continuous, just as we think of material that way, though we know material is generally discrete, what with the knots in the timber, the fibres in the fibreglass, and of course the molecules and the atoms in everything. If time were discrete, then life would be a three-dimensional cine film, and we would live a little jerkily from one moment to the next. Each moment would be a three-dimensional frame. Figure 1 shows two alternative ways we could number these frames.

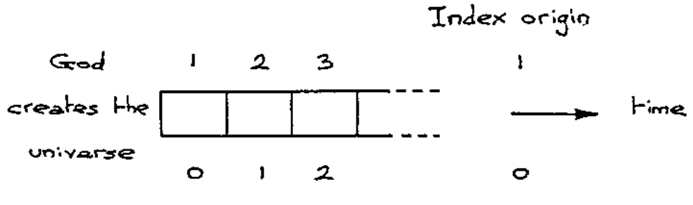


Figure 1

This figure is drawn as if God created time with the universe. Suppose he didn't. Suppose he had already created time, then we have frames before creation as well as after, and it would be natural to number the frames before creation in exactly the same way as those we have already numbered and to distinguish them from each other by a mark as shown in fihure 2.

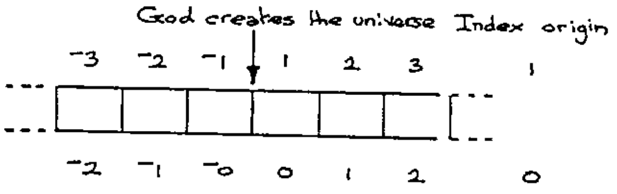


Figure 2

When did God create the universe? A difficult question to answer, because he did it in between moments, so we need a numbering scheme for the divisions between moments. What numbering scheme should we choose? To preserve our symmetry we need a special name for the creation of the universe, which I shall call O (for Origin), but then the names we used for the moments can be used for the divisions as well. This is shown in figure 3.

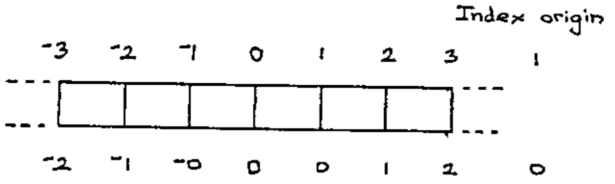


Figure 3

I think I shall use index origin as 1, and argue that zero is very special, rather like infinity. Historically, zero like infinity is a late arrival on the mathematical stage. The

Christian calendar is based on the origin one option of figure 2. There is no year zero. Your ancestors born in year 1 BCD had their first birthdays in 1 AD, though of course they didn't know this at the time since the Christian calendar was not instituted until the fourth century. I could support 1 as being the first integer because integral means whole and an integer implies a whole number. But there is nothing whole about nothing, so although zero is commonly counted as an integer, it is of dubious pedigree.

It may be that you think I am prejudiced against zero. Let me assure you I am very fond of it, and only wish it its proper place. It doesn't want to be first, not even zeroth!

Let us turn to some very simple functions that are index origin independent and see how important nothing is:

```
      ∇R←NUMBER OF THINGS
[1]   R ← NUMBER ρ THINGS ∇

      4 OF 1
1 1 1 1

      3 OF 'X'
XXX

      0 OF 'X'

```

With 0 as the left hand argument, nothing is printed, not even a space or a new line. R in function OF was assigned emptiness, another beautiful concept that Iverson included in APL. The reason I have called OF a counting function is that it shows us what our numbers mean, and introduce us to what might be called the unitary system, even more basic than the binary system, in which we can count I II III IIII IIIII . . . . Notice that the natural zero of this system is emptiness, just nothing, 0 OF 'I'. In the unitary system we can count with just one symbol.

There is an inverse function to OF which I shall call WHAT_NUMBER_OF, which looks like this:

```
      ∇R ← WHAT_NUMBER_OF THINGS
[1]   R ← ⍴ THINGS ∇

      WHAT_NUMBER_OF 1 1 1 1 1
5

      WHAT_NUMBER_OF 'XXX'
3

      WHAT_NUMBER_OF ''
0
```

See that I have represented nothing by two consecutive quotes, which is reasonable enough. Another way of defining emptiness in APL is by using the function iota with zero argument, iota-0. Even though iota is generally index origin dependent, iota-0 always results in emptiness. Emptiness is very useful in APL for avoiding giving special treatment to what in other languages would be special cases.

Finally, I make a plea to everyone not to number beginning at 0. There are one or two exceptions: revision 0 is acceptable if it refers to the unrevised edition, i.e. revision 0 is not a revision. Similarly chapter 0 could possibly be a name for the contents. When counting the steps in a staircase, Richard Smith, the 4 year old son of the editor of this Vector, rightly stamps his foot on the ground with the shout of nought before counting 1, 2, 3, . . as his feet meet each of the steps. The count of nought denoted "this is not a step"; it was special, it denoted "I'm beginning to count now".

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

GENERAL ARTICLES

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

Teaching Statistics in University College, Swansea

by A.D. Mayer and A.M. Sykes

1. Introduction

The teaching of statistics may be approached in a variety of ways, from purely theoretical to strictly applications-oriented. In schools and colleges, the application of statistical techniques to "real" data has always posed a problem. Courses using only calculators give valuable experience in the manipulation of data, but each application tends to be very time-consuming. On the other hand, the use of computers frequently leads to a course which teaches more computing than statistics. Ideally, computers should be used to provide an environment in which statistics can be taught effectively, without demanding any computing expertise.

Specifically, it is desirable that data should be made available to students, to be investigated, displayed and manipulated. Arithmetic calculations must be carried out in an interactive environment, where results can be obtained rapidly. Repetition of commands should be avoided: the ability to define "macros" is essential, especially for complex but frequently used calculations. At least a rudimentary facility for graphs, histograms etc. is needed, so that data can be displayed in a variety of ways, as rapidly as possible. Also required is the generation of random numbers from various distributions for the purpose of simulation.

As in all teaching environments, the facility for students to explore data and methods for themselves, without unnecessary guidance, is highly desirable. Some guidance is, however, essential, and it is helpful if the nature and timing of guidance can be determined by regular assessment of progress. Traditionally, assessment means a degree of individual attention which may be impractical on a regular basis for large classes. Partial automation of this aspect of teaching would be an attractive proposition.

The Department of Management Science and Statistics at Swansea University typically has a class of over 120 first-year students. In recent years, we have tried to develop a sound environment for supporting theoretical and practical studies in elementary probability and statistics. In the following sections we discuss some of the problems outlined above together with our solutions.

2. A Statistical Conversational Environment

Experienced APL programmers rightly consider the concise mathematical APL notation to be one of the most attractive features of the language. It is also the one factor which deters many people from learning APL.

To help users get to grips with the APL environment in their early days, a set of simple functions has been developed to provide "English" commands. Some students transfer very rapidly from these commands to symbolic notation; some take longer; a few never make the transfer. We have to remember at all times that teaching statistics comes first, not training students to be APL programmers.

As an example of our approach, APL commands such as

```
        ROW3 ← MATRIX[3;]
        COL5 ← MATRIX[;5]
```

are made "conversational" by using two functions ROW and COLUMN:

```
        ROW3 ← MATRIX ROW 3
        COL5 ← MATRIX COLUMN 5
```

The gain may appear small, and possibly outweighed by the extra typing involved, but most of the first-year students, for whom computing is a new experience, are far happier with a machine that speaks (and understands) their language.

The following examples typify this approach.

MEANING	APL	FUNCTION
No.of ways of choosing R items from N	R!N	N CHOOSE R
e^x	*X	EXP X
$\log_e x$ or ln x	⍟X	LN X
Maximum of the components of x	⌈/X	MAX X
Minimum of the components of x	⌊/X	MIN X
Sum of the components of x	+/X	SUM X
Logical x OR y	X ∨ Y	X OR Y
Logical x AND y	X ∧ Y	X AND Y
Selection	(X=3)/X	(X=3) SELECT *X*
Order the x's (increasing)	X[⍋X]	ORDERUP X
Order the x's (decreasing)	X[⍒X]	ORDERDOWN X

3. Arithmetic Progressions

The monadic use of Iota occurs very often in APL programming and not surprisingly in Statistics. We have found the function TO which extends the essential idea of Iota to cope with starting points other than 1 (or 0) even more useful. For example 2 TO 5 yields the vector 2 3 4 5, being equivalent to

```
                    (I-⎕IO)+⍳1+J-I      where I=2 and J=5.
```

A further extension using function STEP allows non-consecutive sequences of integers to be constructed. For example

```
                MEAN X[2 TO N STEP 2]
```

calculates the sample mean of all the even values X[2], X[4],...,X[N].

Functions TO and STEP are listed in the appendix.

4. Descriptive Statistics

Most Statistics courses begin with elementary descriptive statistics of single samples, such as mean, standard deviation, etc. Whilst it is tempting to show students how easy it is to calculate such quantities without writing programs, we prefer to provide them with the tools, and allow them time to use them intelligently to gain statistical experience.

As well as MEAN (mean) and SDEV (standard deviation), PCTILES allows them to calculate percentiles and hence to construct alternatives to mean and standard deviation, and the function STATS gives a useful summary. With such functions available students can be presented with data where they must choose carefully whether to use mean or median, standard deviation or semi-interquartile range.

For example,

```
      MEAN AIRPOL
4.4138

      SDEV AIRPOL
1.554
      25 50 75 PCTILES AIRPOL
3 4.65 5.25

      STATS AIRPOL

      MINIMUM               =     2.1
      LOWER QUARTILE        =     3
      MEDIAN                =     4.65
      UPPER QUARTILE        =     5.25
      MAXIMUM               =     8.6
      MEAN                  =     4.41
      STANDARD DEVIATION    =     1.55
      SEMI I-Q RANGE        =     1.125
```

(N.B. the SEMI IQ RANGE is equal to .5X-/25 75 PCTILES X.)

5. Expected Values

The mathematical notation for the expected value of a discrete random variable X taking values x_1, x_2, ..., x_n with probabilities p_1, p_2,...,p_n is given by

$$E(X) = \sum_{i=1}^{n} p_i x_i$$

The omission of any explicit reference to the p_i's in the left-hand side is a source of confusion to students and shows that mathematical notation is not always as precise as mathematicians think it is. At the APL terminal our students improve the notation by typing statements such as

```
M ← P E(X)
```

to assign the expected value (or mean) to M. (The brackets are of course unnecessary if a space is left but we adopt this approach to reinforce the mathematical notation!)

The function E is simple:

```
      ∇R←P E X
  [1] R←(,P)+.×,X
      ∇
```

but its scope is far reaching - for example

```
      P E(X-M)*2
```

calculates the theoretical variance. (For a more detailed explanation of this function see [1].)

6. Displaying Data

Visual representation of data, such as graphs and histograms, are invaluable, particularly for students working with real data for the first time. Initially, we supply such facilities using ordinary (non-graphics) terminals. Later on in the year they are introduced to a graphics workspace that includes the same functions with identical syntax, but which results in a picture drawn on a Hewlett Packard graph plotter. The basic functions supplied are PLOT, HIST, BTBHIST, BOXPLOT, MANYBOX.

For example, the program HIST produces a histogram of the data, but the user must choose the class-width. So an intelligent user of HIST on AIRPOL might try

```
      2  2  HIST AIRPOL

      3  |  ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
      5  |  ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
      7  |  ⎕⎕⎕⎕
      9  |  ⎕
```

and decide that class width 2 is too large.

```
      2  .5  HIST  AIRPOL

      2.25  |  ⎕⎕⎕⎕⎕
      2.75  |  ⎕⎕⎕⎕⎕
      3.25  |  ⎕
      3.75  |  ⎕
      4.25  |  ⎕⎕⎕⎕⎕
      4.75  |  ⎕⎕⎕⎕⎕⎕⎕⎕⎕
      5.25  |  ⎕⎕⎕⎕
      5.75  |  ⎕
      6.25  |  ⎕⎕
      6.75  |  ⎕
      7.25  |
      7.75  |  ⎕
      8.25  |
      8.75  |  ⎕
```

shows that .5 is too small.

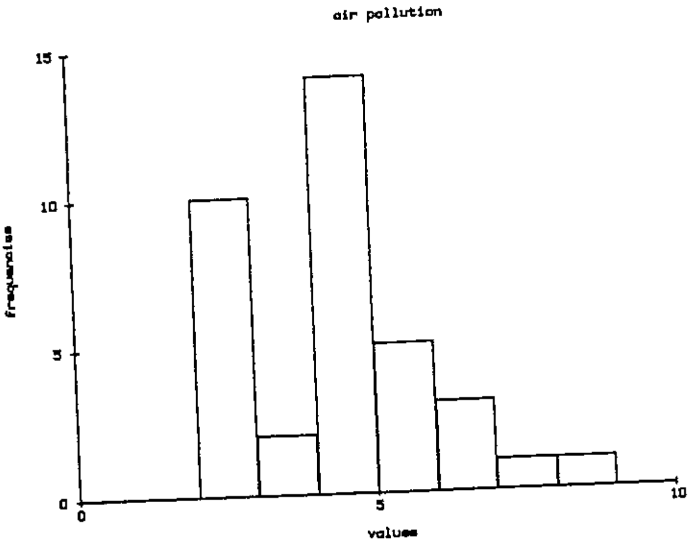
```
      2 1  HIST AIRPOL

      2.5   | ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
      3.5   | ⎕⎕
      4.5   | ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
      5.5   | ⎕⎕⎕⎕⎕
      6.5   | ⎕⎕⎕
      7.5   | ⎕
      8.5   | ⎕
```

looks about right!

No facility for calculating "reasonable" values of the parameters for the data is supplied. The omission is intentional, as the trial-and-error approach, together with a close examination of the data, is very instructive and must be encouraged. Too much "automation" leads to total dependence on the computer, and very little insight into the structure of data.

The HP Plotter version gives:

```
      2 1 HIST AIRPOL
```



The function BTHIST exploits the approach adopted by HIST to produce Back-to-Back histograms to allow contrast of two data vectors. For example, in their second practical, students would contrast the distribution of male weights and female weights in a database of 100 student records as follows.

```
        WEIGHT ← STUDENT COLUMN 4
        SEX ← STUDENT COLUMN 2
        MWEIGHT ← (SEX = 1) SELECT WEIGHT
        FWEIGHT ← (SEX = 2) SELECT WEIGHT

        MWEIGHT BTHIST FWEIGHT

Enter the LOWER BOUND and CLASS WIDTH
⎕:
        30 5

                32.5 *
                37.5
                42.5
             *  47.5 **********
          ****  52.5 ***************
        ******  57.5 **********
   ***********  62.5 *****
**************  67.5 *****
  ************  72.5
           ***  77.5
             *  82.5
                87.5
             *  92.5
             *  97.5
```

With the relatively recent emergence of Tukey's (see [5]) BOX and WHISKER, and SCHEMATIC PLOTS, BOXPLOT attempts to summarise the main features of a histogram, having a central box to represent the three quartiles, with whiskers out to the first 'fence' and two classes of outliers represented by the 'jot' and 'star' symbols respectively.

For example

```
         60 BOXPLOT ?50⍴50

             --------------------------------------
|-----------|                    |                 |---------|
             --------------------------------------

↑                                                            ↑
3.000                                                   49.000
```

yields a box of width 60 characters representing the salient features of the random set of data.

Such BOX or SCHEMATIC PLOTS can be applied to more than one set of data, by allocating each data vector to Y1, Y2 and so on. For example

```
        Y1 ← MWEIGHT
        Y2 ← FWEIGHT
```

```
      50 MANYBOX 2

                           ------
              |----------|  |  |----------|       o   o
                           ------

                  ------
   o         |---|  |  |--------|
                  ------

   ↑                                              ↑
   33.000                                      96.000
```

7. <u>Exploring the Environment</u>

During the course, functions are introduced a few at a time, as required to solve problems derived from lectures. When the workspace is loaded, a description of the latest functions appears. Thus the student is guided to those functions most suitable for the current topic. However, all functions are included in the workspace from the beginning of the course, so that students are encouraged to explore and learn for themselves.

To aid this, a HELP facility is provided. Typing HELP 'function-name' yields a brief description in the format

```
FUNCTION NAME

Description

Right argument
Left argument (if required)

Example

Notes (e.g. reference to other functions, or APL notation in the
case of functions described in paragraph 2.)
```

For example:

```
          HELP 'BIN'

BIN

CALCULATES THE BINOMIAL DISTRIBUTION PROBABILITIES

RIGHT ARGUMENT:- NUMBER OF TRIALS AND THE PROBABILITY(IES)
LEFT ARGUMENT:- OPTIONAL, THE TERMS REQUIRED

EXAMPLES         P←BIN 2 .5
                 P
.25 .5 .25

                 BIN 2 .5 .6
.25 .5  .25
.16 .48 .36
                 (5 TO 7) BIN 10 .5
.246 .205 .117

N.B. A BAR CHART OF THE PROBABILITY DISTRIBUTION CAN BE PRODUCED USING
BAR AND DIST. E.G.
       P←BIN 10 .5
       V←0 TO 10
       BAR P DIST V
```

The HELP facility works by accessing numbers inserted as a comment on the first line of every function, indicating which component of the help file is to be displayed. The HELP function as implemented in APLPLUS is listed in the appendix.

8. Assessment of Students

Periodically throughout the course, students are asked to perform tasks and submit their answers for computer marking. Their marks contribute about 20% to their total mark at the end of the year, ensuring that they take their computer practical sessions seriously!

The basic steps in their continuous assessment is as follows:-

```
)COPY 0:ASSESS           (copy in the current assess workspace)
TASK                    (creates question and gives instructions)
  .
  .                     (APL session)
  .
ANSWER ←                (allocate answer to variable)

SUBMIT                  (erases all APL objects except ANSWER and
                         saves in workspace called ANSWERS)
```

The function TASK itself involves functions that perform the following tasks:

(i) set the random link according to the user number
(ii) generate the question
(iii) issue message.

Functions (ii) and (iii) are up-dated for each assignment, otherwise the ASSESS workspace remains the same each week.

The supervision of computer assignments needs a workspace to monitor the work submitted and to assign marks. The functions necessary for this are

(1) GETANSWER USERNAME

This copies the student's answer into the workspace.

(This is so arranged that if it fails to locate an answer, it returns a default vector).

(2)	Q←GETQUESTION USERNAME	(a version of TASK)
(3)	TA←TRUEANSWER Q	(calculate correct answer)
(4)	M←TA MARK A	(marks by comparing answer A with true answer TA)

All four functions form the core of a drive function which loops through a list of usernames, with each result M of function 4 being inserted into a vector of MARKS.

Here is an example of a sample session, where a student is presented with a data set of between 15 and 20 data items generated (possibly) from a Normal distribution with a hypothesised mean of 20, which they have to test in an appropriate way. Note that all students get different data, and they cannot assume that Jo Bloggs' answer is appropriate for them too!

```
      TASK

THE FOLLOWING DATA IS A RANDOM SAMPLE FROM A POPULATION WITH
ASSUMED POPULATION MEAN=20. TEST THIS HYPOTHESIS USING A SUITABLE
TEST. (IF YOU ARE SATISFIED THAT THE DATA HAS A NORMAL DISTRIBUTION
THEN USE A T TEST. OTHERWISE USE THE WILCOXON RANK SUM TEST AS
DESCRIBED IN LECTURES.)
YOUR ANSWER SHOULD BE A THREE ELEMENT VECTOR
FIRST ELEMENT =1 IF YOU HAVE ASSUMED A NORMAL DISTRIBUTION, OTHERWISE
0. SECOND ELEMENT=TEST STATISTIC, THIRD ELEMENT=(TWO-SIDED)
SIGNIFICANCE LEVEL

ASSIGN THESE THREE VALUES TO A VARIABLE CALLED ANSWER AND TYPE SUBMIT

THE DATA IS AVAILABLE UNDER THE NAME DATA
27.55 22.45 18.45 25.36 13.47 28.35 20.89 18.42 6.48 22.69 19.24 24.18
      27.70 19.82 37.33 17.50 42.03

      )COPY 0:STATS
SAVED 11-JAN-1987 15:43:33 119 BLKS

      DATA
27.55 22.45 18.45 25.36 13.47 28.35 20.89 18.42 6.48 22.69 19.24 24.18
      27.70 19.82 37.33 17.50 42.03

      STATS DATA
MINIMUM            = 6.48
LOWER QUARTILE     = 18.44
MEDIAN             = 22.45
UPPER QUARTILE     = 27.58
MAXIMUM            = 42.03
MEAN               = 23.05
STANDARD DEVIATION = 8.33
SEMI I-Q RANGE     = 4.57

      5 10 HIST DATA

10 | ⎕⎕
20 | ⎕⎕⎕⎕⎕⎕⎕⎕⎕
30 | ⎕⎕⎕⎕
40 | ⎕⎕

      5 5 HIST DATA
 7.5 | ⎕
12.5 | ⎕
17.5 | ⎕⎕⎕⎕⎕
22.5 | ⎕⎕⎕⎕
27.5 | ⎕⎕⎕⎕
32.5 |
37.5 | ⎕
42.5 | ⎕
```

```
      ⍝DOESN'T LOOK VERY NORMAL
      X←DATA-20
      SUM X=0
0
      R←RANK |X
      S←SUM (X>0)/R
      S
108

      M←.5×SUM R
      V←.25×SUM R×R
      TESTSTAT←S-M
      TESTSTAT←TESTSTAT÷V*.5
      TESTSTAT
1.49
      2×NORMTAIL TESTSTAT
0.14
      ANSWER←0,TESTSTAT,2×NORMTAIL TESTSTAT
      ANSWER
0 1.49 0.14
      SUBMIT
```

As with all the topics discussed in this paper, there is plenty of scope for extension, and our next aim is to extend the ideas embodied in computer assessment to construct a body of questions when students could use to self-assess; each type of question can be repeated as many times as required, and students have access to written solutions, plus an assessment of their marks, either for a particular question, or for all the questions selected. It is hoped that this will be in operation later on this year.

9. Databases

One advantage of our approach is that realistic examples of data can be tackled just as easily as textbook-size examples. A number of databases have been collected and are available to the student for practical use. Each database also contains information showing how the database is constructed.

Each year students fill in a questionnaire (at the computer of course) and this is assembled to form a further database which they subsequently analyse.

10. Summary

The statistics workspace used by part one students allows them to concentrate their energies on doing and understanding statistics. In so doing however, they inevitably pick up some of the flavour of APL and those that are at all interested are able to experiment to their hearts content. Those that find it difficult are at least encouraged initially because of the lack of APL characters (only the use of '←' is necessary together with +,-,¯,X,+ and * for the elementary arithmetic operators), and by the presence of the HELP instructions which guide them in using the facilities.

In the second year, an option in Statistical computing allows the keen ones to study the language in its entirity and includes projects in statistical computing which require them to find solutions to statistical computing problems. Others who continue with their statistical studies have sufficient command of APL to be able to use more advanced statistical software such as APLGLIM (see [2], [3] and [4])

11. Appendix A

6. Functions in the STATS workspace

There are 6 types of functions

(a) Functions performing ordinary data manipulative tasks;

(b) Elementary statistical calculations;

(c) Character graphics facilities;

(d) Statistical table functions

(e) More advanced statistical calculations

(f) Specific teaching aids

and these groups are listed below.

Group a	AND CHOOSE CODE COLUMN COUNT CUMSUM DROP EXP LN MAX MIN OR ORDERDOWN ORDERUP ROW SELECT SHAPE SUM SUMCOL TAKE TO STEP
Group b	FTABLE GMEAN GSDEV MEAN MEANS PCTILES SDEV STATS STEMLEAF TWTABLE RANK UNIQUE
Group c	BAR BOXPLOT BTBHIST HIST MANYBOX PLOT QQPLOT
Group d	BIN CHISQUARE FTAIL NORMQUANTILE NORMTAIL TTAIL
Group e	ANOVA CONTINGENCY NPTWOSAM TWOSAMT WILCOXON
Group f	e.g. CONFTRIAL DICE DIST E NRSAMPLE RSAMPLE SAMPLE TIMES

12. Appendix B

Some listings of functions in 0:STATS as mentioned in the text.

```
       ∇PLOT[⎕]∇
 [0]  R←Y PLOT X;MIX;MAX;MIY;MAY;L1;L2;D;T
 [1]  ⍝015
 [2]   →(0≠⍴R←(33×(⍴X)≠⍴Y)⍴'ARGUMENTS HAVE UNEQUAL DIMENSIONS')/0
 [3]   MIX←⌊/X ⋄ MAX←⌈/X ⋄ MIY←⌊/Y ⋄ MAY←⌈/Y
 [4]   →(0≠⍴R←(16×1=(MIY=MAY)∨MAX=MIX)⍴'DEGENERATE RANGE')/0
 [5]   Y←⌊1.5+19×(Y-MIY)÷(MAY-MIY)
 [6]   X←⌊1.5+29×(X-MIX)÷(MAX-MIX)
 [7]   R←⊖⍉(⍉X∘.=⍳30)+.×Y∘.=⍳20
 [8]   R←(60⍴ 1 0)\' ○⊖⍟'[4⌊1+R]
 [9]   L1←⍕MAX ⋄ L2←⍕MIX
 [10]  R←('|',R),[1]'¯'
 [11]  R←R,[1]L2,((61-(⍴L1)+⍴L2)⍴' '),L1
 [12]  D←8⌊(⍴⍕MAY)⌈⍴⍕MIY
 [13]  T←(22,D)⍴' '
 [14]  T[1;]←D↑⍕MAY ⋄ T[20;]←D↑⍕MIY
 [15]  R←T,R

       ∇HIST[⎕]∇
 [0]  T←C HIST X;F;R
 [1]  ⍝010
 [2]   T←BAR C FTABLE X
       ∇BAR[⎕]∇
 [0]  R←BAR T;F
 [1]  ⍝001
 [2]  ⍝ TAKES A FREQUENCY TABLE OR DISTRIBUTION TABLE AS RIGHT ARGUMENT
 [3]  ⍝ USE FTABLE OR DIST
 [4]   R←T ⋄ →(2≠⍴⍴R)/0
 [5]   →(1≠+/,T[;2])/FR ⋄ T←T×(⍴T)⍴ 1 80
 [6]  FR:R←' ⎕'[1+⍉(⍳⌈⌈/F)∘.≤F←T[;2]]
 [7]   ' '
 [8]   R←(⍕ 0 ¯1 ↓T),(((⍴F),3)⍴' | '),R
       ∇FTABLE[⎕]∇
 [0]  R←C FTABLE X;N
 [1]  ⍝008
 [2]  ⍝LEFT-HAND ARGUMENT IS MINIMUM,CLASS-WIDTH
 [3]  ⍝RIGHT-HAND ARGUMENT IS RAW DATA
 [4]   →(C[1]≥⌊/X)/ER
 [5]   N←⌈((⌈/X)-C[1])÷C[2]
 [6]   R←(C[1]+C[2]×¯0.5+⍳N),[1.5],+/(⍳N)∘.=⌈(X-1↑C)÷C[2]
 [7]   →0
 [8]  ER:R←'SPECIFIED LEFT-HAND POINT DOES NOT INCLUDE ALL THE DATA'
```

```
      ∇BTBHIST[⎕]∇
[0]  X BTBHIST Y;C;T1;T2;M;R1;R2;F
[1]  ⍝002
[2]  ⍝LEFT AND RIGHT ARGUMENTS ARE THE TWO DATA VECTORS
[3]   'ENTER THE LOWER BOUND AND THE CLASS WIDTH'
[4]   C←⎕ ⋄ T1←C FTABLE X ⋄ →(2≠⍴⍴T1)/er
[5]   T2←C FTABLE Y ⋄ →(2≠⍴⍴T2)/er
[6]   M←(⍴T1)⌈⍴T2 ⋄ T1←M↑T1 ⋄ T2←M↑T2
[7]   T1[;1]←T1[;1]+T2[;1]-T1[;1]×T1[;1]=T2[;1]
[8]   R1←' *'[1+⍉(⍳⌈/F)∘.≤F←T1[;2]]
[9]   R2←' *'[1+⍉(⍳⌈/F)∘.≤F←T2[;2]]
[10]  2 1 ⍴' '
[11]  ((⊖R1),' '),((⍕ 0 ¯1 ↓T1),(((⍴F),1)⍴' ')),R2
[12]  2 1 ⍴' '
[13]  →0
[14] er:'SPECIFIED LEFT-HAND POINT DOES NOT INCLUDE ALL DATA'

      ∇BOXPLOT[⎕]∇
[0]  R←WD BOXPLOT D;L;U;A;F
[1]  ⍝074
[2]   ' ' ⋄ L←⌊/D ⋄ U←⌈/D
[3]   P←WD ∆BOXPARAM L,U,D ⋄ D←13↓P ⋄ P←11↑2↓P
[4]   R←D ∆BOX WD,P
[5]   R←R,[1](2,1↓⍴R)⍴' ' ⋄ R[5;1,1↓⍴R]←'↑'
[6]   R←R,[1]' '
[7]   A←,'F8.3' ⎕FMT L ⋄ A←(' '≠A)/A ⋄ R[6;⍳⍴A]←A
[8]   A←,'F8.3' ⎕FMT U ⋄ R[6;((1↓⍴R)-⍴A)+⍳⍴A]←A
[9]
      ∇MANYBOX[⎕]∇
[0]  B←WD MANYBOX NUM;S;C;L;U;D;P;Y
[1]  ⍝075
[2]   ' ' ⋄ L←⌊/Y1 ⋄ U←⌈/Y1 ⋄ C←1
[3]  lp:L←L⌊⌊/Y←⍎'Y',⍕C ⋄ U←U⌈⌈/Y
[4]   →(NUM≥C←C+1)/lp
[5]   B←(0,WD+1)⍴' ' ⋄ C←1
[6]  lp2:D←⍎'Y',⍕C ⋄ P←WD ∆BOXPARAM L,U,D
[7]   D←13↓P ⋄ P←11↑2↓P
[8]   B←B,[1]' ' ⋄ B←B,[1]D ∆BOX WD,P
[9]   →(NUM≥C←C+1)/lp2
[10]  B←B,[1](3,1↓⍴B)⍴' ' ⋄ S←1↑⍴B ⋄ B[S-1;1,WD+1]←'↑'
[11]  Y←,'F8.3' ⎕FMT L ⋄ Y←(' '≠Y)/Y ⋄ B[S;⍳⍴Y]←Y
[12]  Y←,'F8.3' ⎕FMT U ⋄ B[S;(WD+1-⍴Y)+⍳⍴Y]←Y
[13]
      ∇∆BOX[⎕]∇
[0]  R←D ∆BOX P;WD
[1]   WD←1↑P ⋄ P←1↓P ⋄ R←(3,WD+1)⍴' '
[2]   R[2;P[4 5 6 7 8]]←'⊢|||⊣'
[3]   R[2;P[4]+⍳¯1+-/P[5 4]]←'-'
[4]   R[2;P[7]+⍳¯1+-/P[8 7]]←'-'
[5]   R[1 3 ;P[5]+⍳¯1+-/P[7 5]]←'-'
[6]   R[2;((D<P[3])∨D>P[9])/D]←'∘'
[7]   R[2;((D<P[2])∨D>P[10])/D]←'*'
      ∇∆BOXPARAM[⎕]∇
[0]  P←WD ∆BOXPARAM D;Q;L;U;IF;OF;LA;UA
[1]   L←D[1] ⋄ U←D[2] ⋄ D←2↓D ⋄ D←D[⍋D] ⋄ Q← 25 50 75 PCTILES D
[2]   IF←Q[1 3]+.× 2 2 ⍴ 2.5 ¯1.5 ¯1.5 2.5
[3]   OF←Q[1 3]+.× 2 2 ⍴ 4 ¯3 ¯3 4
[4]   LA←1↑(D≥IF[1])/D ⋄ UA←¯1↑(D≤IF[2])/D
[5]   P←L,OF[1],IF[1],LA,Q,UA,IF[2],OF[2],U
[6]   P←⌈0.5+WD×(P-L)÷U-L ⋄ D←⌈0.5+WD×(D-L)÷U-L
[7]   P←L,U,P,D
```

```
      ∇HELP[⎕]∇
[0]  HELP N;I;K;⎕ELX
[1]  ⍝009
[2]   ⎕ELX←'→er'
[3]   'HELPSTAT' ⎕FTIE 1
[4]   N←⍎(⎕CR N)[2; 2 3 4]
[5]   ' ' ⋄ ⎕FREAD 1,N ⋄ ' ' ⋄ ⎕FUNTIE 1 ⋄ →0
[6]  er:'HAVE YOU USED HELP WITH A VALID FUNCTION NAME?'
[7]   ⎕FUNTIE 1


      ∇TO[⎕]∇
[0]  R←I TO J
[1]  ⍝028
[2]   J←J,1
[3]   R←(I-⎕IO)+J[1+⎕IO]×⍳⌈(1+J[⎕IO]-I)÷J[1+⎕IO]
      ∇STEP[⎕]∇
[0]  R←Y STEP Z
[1]  ⍝029
[2]   R←Y,Z
```

References

[1] HAWKES A.G. & SYKES A.M.
Teaching Statistics through APL.
Tutorial Volume APL '86.

[2] SYKES A.M. & ANSELL J.I.
Applied Linear Interactive Modelling.
Proceedings of APL '85.

[3] SYKES A.M.
Generalized Linear Models in APL.
Proceedings of APL '86.

[4] SYKES A.M.
Second Generation Domino for Statisticians.
Proceedings of APL '87.

[5] TUKEY J.W.
Exploratory Data Analysis.
Addison-Wesley (1977).

The I-APL Project: History and Achievements

by Anthony Camacho

Paper given in the Education Stream of the
Danish APL Association Meeting September 1987

Printed by permission of the APL group of the Dansk Databehandlingsforening

Introduction

Those who have had to teach subjects, such as maths stats and physics, involving many mathematical evaluations, have regretted that computers have been so little used in teaching. Those of us who know about APL are always surprised how little it is used in schools because it seems to overcome some of the difficulties admirably.

Paul Chapman added another ingredient to this dissatisfaction and surprise - the proposal that it would be possible to write an APL interpreter small enough to run on school and home machines. This addition catalysed a reaction and at APL 86 a committee was elected to raise money and execute the project. That was in July 1986 and as you know execution is a slow function. We have just (Sep 87) begun some beta-testing. This paper is to tell you something about the I-APL project, its history and the I-APL interpreter which it has produced.

The fund raising

I don't want to say anything about this except that it was difficult: we are still very short of money and will need to pay for the printing of manuals soon. Any contribution you can make will be very welcome.

The Organisation

Five people were elected to the original committee: Ed Cherlin and Anthony Camacho to be a chairman each side of the Atlantic, Howard Peelle and Norman Thomson to manage education each side of the Atlantic and David Ziemann to act as technical controller. Dave was at school with Paul Chapman and has been friends with him for many years. A Limited Company was formed in the UK with these five people as directors. We did this because a company has to report its accounts each year and we wanted to give contributors confidence that their money would be well and publicly accounted for. In addition to the committee Garry Helzer, Linda Alvord, Sylvia Camacho and many other people have given a great deal of time to the project.

Objectives

The five directors declared that the object is to write and issue, together with appropriate supporting documentation and workspaces, an APL interpreter for which there will be no charge. The project may have to charge for copying and materials but prices will be judged to break even rather than make a profit. The directors have undertaken never to pay any fees to directors nor to pay any dividends to shareholders. Most of the expenses that people have incurred have not been claimed. Any surplus funds are to be used for the promotion of APL. Note that the project is to promote APL - not I-APL which is just a means to the end and not an end in itself.

Free, standard and without commercial bias

One of the obstacles which deter educational establishments considering the use of APL is that every version of APL is a proprietary product and it would be wrong for a school to promote a particular vendor. None of the vendor APLs completely conforms to the ISO standard. It was a piece of good luck for us that the standard was in draft when we began and that we were able to make I-APL conform to it.

We decided that not having a vendor (because the product is free) was a great advantage which we could double by sticking to the ISO standard. Paul Chapman used the standard as his guide during the design of the primitive functions. He found several places where the standard was not well expressed but as David Ziemann is chairman of the ISO APL committee the problems were solved.

I-APL can be used with current introductions to APL. An APL which would be free, standard-conforming and impartial between vendors would be ideal for the educational world but if that meant it would have to begin life without being able to use the current world stock of APL texts and software nobody would be able to use it. Many of the best texts use direct definition: the ISO standard includes the Del editor. We felt we had to implement both. Dave Ziemann has started on the road to establish compatibility by writing a workspace which transfers APL*PLUS/PC workspaces to I-APL/PC form.

Product documentation

We also needed documentation to go with the product. A reference manual, an introduction to APL and an installation manual seemed to us to be the minimum. The first two could apply to any version of APL. Everything which was specific to I-APL or to the machine implementation could be put in the installation manual. Garry Helzer has written the reference manual. It is called "An APL Encyclopaedia". Norman Thomson has written the APL Tutorial, using some material supplied by Linda Alvord. Both these are finished and almost in camera ready form. All we have to do is find a publisher or pay to print them ourselves.

The installation manual

The third text is being worked on. To be acceptable to people who have become used to the facilities of their machine, any new piece of software must give access to the graphics, colour and sound of the native machine, allow printing of APL on the common printers and provide means for owners of uncommon printers to patch their machine so that even they can be used. This work cannot be done before the beta testing: the last part of the manual writing cannot be done until this work is finished. For this reason we expect not to issue any free APLs before December at the earliest.

Messages in any language

To justify the slogan "The Free International APL" we had to build the interpreter in such a way that it would be easy to change the messages into any other language. Our original method caused some difficulty for porters producing versions for languages which had word orders different from English. Paul has revised this part of the interpreter with Kim Andreasen's advice and there should now be no difficulty in producing all the messages in any language which uses the roman alphabet.

Easy porting

To make the interpreter easy to port, Paul decided to write the APL in an intermediate language. This language was to be as simple as he could make it and so would not require much work to implement or much space in the machine. The language is called DE which you may think of as standing for Development Environment or as being two better than 'C'. The I-APL interpreter is written in DE and compiled into a fixed final form which will be exactly the same in all machines (a file of a little over 25K). It passes all messages to the screen or printer via the DE interpreter which nowadays is usually called DEGO. The double interpretation, once as DE and once as APL inevitably means that I-APL is slow. However to port it to another machine all that is required is an appropriate DEGO. As we now have versions of the interpreter running on the 8086, 80186, 6502, Z80 and 68008 CPU chips the amount of work to be done for another machine using one of these chips is quite small. The largest task is to rewrite that part of DEGO which handles the links with the native operating system.

Workspaces are compatible between all I-APL machines

Another consequence of building I-APL as a fixed file interpreted by a DEGO is that the fixed file controls all the formats of the objects and of the whole workspace. All workspaces)SAVEd in one version of I-APL can be)LOADed into any I-APL. There is only one single machine dependent function. It is called quad-MC and is the machine code call. If a workspace does not use this function then it will run and produce the same results on any machine. We believe that this will be particularly valuable in education because if a school in Italy produces some good geaography workspaces they can easily be used in Japan or anywhere else even if the machines there are totally different.

Cover functions for graphics and colour

Educational workspaces would benefit greatly from imaginative use of colour and graphics and it seemed a pity that we could not find a way to help transfer of good work between incompatible machines. Paul hopes to find time to specify an approach to doing graphics and colour work in I-APL so that even if quad-MC is used to produce pictures, the graphic functions for machine 1 can be erased from the workspace and replaced by the graphic functions for machine 2. When this is done the workspace should run giving the nearest that machine 2 can manage to machine 1's display. We do not expect to have this facility ready this year (1987).

The features of I-APL

The demonstration of any good APL interpreter should be boring because everybody knows what it should do. Demonstrations of I-APL are now beginning to be as boring as they should be. It is a complete ISO-conforming APL. It has 21 scalar dyadic, 13 scalar monadic, 22 dyadic mixed and 11 monadic fixed functions. It has three dyadic operators and four monadic operators. Quad and quote-quad can both be used for input and output. It includes indexing and indexed assignment, a suitable selection of system commands and system variables and functions. The main enhancements are replicate and the direct definition feature.

Replicate

In standard APLs the left argument of compress must be a boolean with as many values as the appropriate dimension of the right argument. The result consists of just those items from the right argument corresponding to the ones in the boolean. Replicate allows integers as well as booleans in the left argument and gives as many copies as the left argument requires of the selected items from the right argument.

Direct definition

All directly defined functions are "ambiguous": they may have one argument or two. When a directly defined function is called it must be given a right argument. Typing the function name alone followed by <RETURN> commands the display of the function, so reference to the name alone without arguments is the equivalent of trying, within a function, to enter function definition mode. If the expression(s) within the function contain no reference to the left argument it need not be supplied. A right argument must always be supplied although it need not be used in the expression evaluated.

There are two forms of direct definition. One includes a condition which can be used to select one of two expressions to execute, the other does not and consists of a single expression. The simple form consists of a name, a colon and an expression. In the expression alpha stands for the left argument and omega for the right argument. If any assignments are made in the expression the variables assigned are automatically localised. The result of the function is the result of whichever expression is evaluated.

Examples

In the examples below I have stuck to APL symbols available in ASCII and used lower case a and w for alpha and omega.

Here is the function LEV which returns its left argument unchanged:

```
        LEV: a
```

Functions can call themselves or other functions. Here is a recursive factorial function:

```
        FACT: 1 : 1<w : w x FACT w-1
```

Functions which contain a condition or test are of the form:

```
    FNNAME: <fail expression> : <test expression> : <succeed expression>
```

The test expression is evaluated first and should give a boolean result. If the result is one then the succeed expression is executed and its result becomes the result of the function. If the result of the condition is zero then the fail expression is executed and gives the result of the function.

Display and editing of directly defined functions

In the display of)FNS (but not of quad-NL 3) the directly defined functions are shown with a colon after their name. To display a direct definition function just type its name and return. The function will be displayed and control will then be returned to the keyboard. To edit a direct definition function type its name and a colon and it will be displayed with the cursor at the end of the line ready for you to amend it. If there is already a direct definition of a function and another definition with the same name is entered it simply overwrites the previous version. Benefits of direct definition

People used to writing long functions with the del editor or a full screen editor find that recasting their thoughts to fit direct definition form is quite an effort. I suggest you try it. You don't need I-APL. In Linda Alvord's book "Probability in APL" or in "A Source Book in APL" (both published by APL Press) there are functions to allow entries to be made using the direct form. When you find a good way of fitting a task into the direct definition form it is extremely satisfying. I suspect that the use of direct definition encourages clearer thinking about the structure of an algorithm than is necessary when it is possible to write long functions with a screen editor. It is possible to write an APL imitation of spaghetti BASIC with a good screen editor but I doubt whether it could be done with direct definition.

Porting

The work is going on at this moment. There are over 25 people at work in USA, UK, Denmark, Canada, Sweden, Australia, Switzerland, Finland and Germany. Prototypes are running on IBM/PC, XT, AT and clones, Piccoline, BBC'B' and Master, CBM64, Amiga, a homebrew Z80, Sinclair QL and ARM or Archimedes. Two of these, the PC clone and the BBC B version are on beta test. We are also working on Sinclair Spectrum, Apple II, Nokia, Microbee, Atari ST, Wicat, Luxor, VAX, Wicat and IBM mainframe.

A Short Demonstration of Direct Definition

by Anthony Camacho

I-APL is an educational APL

Some of the best APL teaching books such as those by Alvord, Thomson and Iverson use direct definition.

I-APL includes direct definition as well as the "DEL" method for defining functions so every APL textbook will work with I-APL.

A short demonstration of direct definition follows. Please ask questions whenever you like.

To define a function directly you must have it all on one line. The line length is set by ⎕*PW* and the maximum is 254. The line must begin with the name of the function followed by a colon, thus:

```
      ODDS:(2|A)/A←⍳2×⍵
```

Inside the function the right argument is always ⍵ and the left argument is always α. The result of the function is the result of the whole expression after the colon. All variables assigned a value in the expression (such as A in ODDS) are localised. ODDS N returns the first N odd numbers.

```
      ODDS: 20
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
```

ODDS is not origin sensitive.

```
      ⎕←⎕IO←0
0
      ODDS: 20
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
```

```
      ⎕←⎕IO←1
1
```

This is the simple form of direct definition. It is very easy to use. Here are some more examples:

```
      AV:(+/,⍵)÷⍴,⍵
```

AV returns the average value of the items in its argument.

```
      AV 1 11 3 9 5 7
6
      AV ⍳99
50

      WHERE: ⍺
```

WHERE is the same as Iverson's LEV. It is obviously dyadic but doesn't use its right argument. By simply returning its left argument it allows us to put an expression on the right of any argument without interfering with it. As the right-hand expression is executed first it can set up conditions to prepare for the left-hand expression. Here is a function called ORIGIN0 (the last character in the name is a zero) to make it easy to test any function in origin zero.

```
      ORIGIN0: ⍎⍵ WHERE ⎕IO←0
      ORIGIN0 '⍳9'
0 1 2 3 4 5 6 7 8
```

ASK gives its argument as a prompt and returns whatever is the keyboard response, in this case I keyed "OK"'return'.

```
       ASK: (⍴P)↓⍞ WHERE ⍞←P←⍵,':'
      ASK 'PROMPT'
PROMPT: OK
OK
```

Next is the traditional IF in direct form:

```
        IF: ω/α
```

COL is Iverson's monadic comma-bar "⍪", which produces a one column table of the items in its argument.

```
        COL:((ρ,ω),1)ρω
        COL ⍳5
1
2
3
4
5
```

This function returns the first N square numbers.

```
        SQU: +\ODDS ω
        SQU 16
1 4 9 16 25 36 49 64 81 100 121 144 169 225 256 324
```

Triangular numbers are even easier. Notice how easy it is to localise the index origin with WHERE.

```
        TRI: +\⍳ω WHERE ⎕IO←1
        TRI 16
1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136
```

I owe this FIB function to Joseph de Kerf. You will find it explained in Vector 3.4 on page 120.

```
        FIB: ⌊.5+(÷R)×(.5×1+R←5*.5)*⍳ω
        FIB 10
1 1 2 3 5 8 13 21 34 55
```

Each example so far has consisted of a single expression. Direct definition has a conditional form which contains three expressions. They are separated by colons thus:

```
        FNNAME:<IF-FALSE-EXPR>:<COND>:<IF-TRUE-EXPR>
```

The middle expression is evaluated first and should return a single boolean. If its result is true then the IF-TRUE-EXPR gives the result of the function; if its result is false then the IF-FALSE-EXPR does so.

For example if we wanted a function like ASK, but amended so that it will only accept a Y or N and return 0 for N and 1 forY, we could write ASKYN.

```
        ASKKYN: R : 2=R←'NY'⍳1↑ASK ⍵,'?' WHERE ⎕IO←0 :
ASKYN ⍵ WHERE ⎕←'YES/NO ONLY'
```

The middle expression sets the index origin to zero, adds a question mark to the prompt and then ASKs it, indexing the first character of the response in 'NY'. If R does equal 2 then the initial character of the response was not found in 'NY' and so the response was not YES or NO and the if-true-expression just displays the additional message 'YES/NO ONLY' and tries again. If R doesn't equal 2 then it must have been 0 or 1 (the other possible results in origin zero) and as in either case R contains the required result - all the work has been done - the if-false-expression can just be R.

```
      ASKYN'ARE YOU STILL LYING'
ARE YOU STILL LYING?: UNFAIR
YES/NO ONLY
ARE YOU STILL LYING?: NOT FAIR
0
```

The example shows the reprompt when the response doesn't begin with Y or N and also that the test is not really adequate; the second response is not equivalent to NO.

So a definition can use other definitions and also can use itself. Functions which call themselves are recursive. Recursion can be used instead of loops to repeat an algorithm as many times as needed. I find it difficult to construct recursive functions which work the way I want them to. To ensure a recursive function ends properly it should refer to itself only in the first or last expression and the conditional expression should switch to execution of the other when the recursion is to be ended.

I-APL has a dynamic stack and a dynamic symbol table so if you make a mistake and set off an unbounded recursion the whole memory will be filled with recursive calls before it dies with a WS FULL.

Here is a function which builds a table from lines entered at the keyboard.

```
      MAT: V OVER MAT ω : 0=ρV←⍞ : 0 0ρ' '
```

The function OVER simply catenates its left argument over its right. It uses R2 which forces its argument to be a rank two table, MAXC to return the width of the wider table and EXD to extend the tables to the same (maximum) width before catenating them on the first dimension.

```
      OVER: (C EXD α),[⎕IO]C EXD ω WHERE C←(α←R2
α)MAXC ω←R2 ω
      R2: (¯2↑1 1,ρω)ρω  ⍝ FORCES ω TO RANK 2
      MAXC: (1↓ρα)⌈1↓ρω  ⍝ MAX OF COLS IN TWO TABLES
      EXD: ((ρω)⌈0,α)↑ω  ⍝ EXTEND TABLE TO WIDTH α
```

Here is a use of it. I will enter three names from the keyboard and terminate the function by a 'return'.

```
      NAMETABLE←MAT 'A'
HANS-DIETER
GERALD
ANTHONY

      NAMETABLE
HANS-DIETER
GERALD
ANTHONY
```

I am displeased by the inelegances of MAT. First it requires an argument (I do not see how you can write a recursive niladic function) which has no effect on its result. Second the if-true-expression seems an inelegant waste of time too as it has no effect either; there are only three rows in nametable.

I would be grateful if someone would show me how to do it better and very grateful indeed if someone would teach me how to think these things out so that I could avoid such inelegance without further help.

After that the last two functions I have to show are easy. They are the paradigm of recursion: FACTORIAL and a recursive Fibonacci series generator.

```
      FACTORIAL: ω×FACTORIAL ω-1 : ω≤1 : 1
      FACTORIAL 9
362880
      FIBS: A,+/¯2↑A←FIBS ω-1 : ω=2 : 1 1
      FIBS 10
1 1 2 3 5 8 13 21 34 55
```

Anthony Camacho   revised 14 December 1987

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 - Direct Definition

by Jonathan Barman and Dave Ziemann

It is generally conceded that the `direct definition' of APL functions is much more elegant than the more traditional methods of defining functions in the APL workspace. Where direct definition is used, it tends to promote the development of short, easy to understand functions which can be used as building blocks to create complex systems.

Direct definition provides a way to associate an APL expression with a name in a single step. In the simplest form, a colon is used to separate the name from the expression, and the letters alpha and omega may be incorporated into the expression to represent the left and right function arguments, respectively. Alternatively, a multiple segment form can be used to define a conditional function, from which arbitrarily complex systems of interrelating functions can be constructed.

Over the years a number of different schemes for the direct definition of APL functions has been proposed. One of the most recent is presented later in this issue of Vector, and the article represents a good starting point for those unfamiliar with the idea.

Looking through the APL conference proceedings over the last few years gives the impression that APL purists and educators always use direct definition, but that APL pragmatists never use it. In fact one can almost classify papers by this method. Does the paper use direct definition? If yes, then it's about APL, if no, then it's an application.

The APL application programmer normally avoids using direct definition, presumably because it is not an integral part of the interpreter. Although it is fairly easy to write a pair of utility functions which will fix and display character vectors in direct definition format, it is just too much trouble to invoke tham all the time, the traditional editor being so readily available. You always have to surround your text with quotes, and at worst the utility functions are not in the workspace when you want them.

I-APL has led the way in changing this state of affairs; the ability to directly define functions in immediate execution mode is built into the interpreter. The mere fact that one can type a short function in direct definition format encourages one to do so. The power of APL as a `super calculator' is greatly enhanced when it is possible to directly define short functions to solve problems, or explore a particular avenue by gradually building on existing definitions.

Direct definition is not just good for teaching or learning APL and mathematics however. Commercial APL users could also benefit by its inclusion, and at no great cost; the minimal direct definition scheme used in I-APL accounts for about 300 extra bytes of interpreter code.

A direct definition scheme need not go against the ISO APL Standard either. In the standard the immediate execution form <name>:<expression> produces an error, and so this error can be replaced by any other behaviour - in this case the definition of a function. In I-APL direct definition is a consistent extension to the APL standard.

We welcome correspondence on direct definition, and your experiences with it.

Hacker's Corner

by Adrian Smith

Rather than upgrading your copy of APL*PLUS/PC to version 4.2, you might find the following locations useful. If you have access to a utility such as PC Tools or 'Norton' you can simple edit the .EXE file directly in hex; otherwise the little function below will do the trick:

```
    CUSTOM BANNER INSTALLATION FOR APL*PLUS/PC
    ==========================================

FI←'ZAP7.EXE'   ⍝ .... set filename to xxxx.EXE

BNR←123940 ⋄ SER←4126 ⋄ VER←100957   ⍝ Set up for Vn5
BNR←140932 ⋄ SER←5150 ⋄ VER←114344   ⍝ Set up for Vn6
BNR←153796 ⋄ SER←6175 ⋄ VER←124716   ⍝ Set up for Vn7

FI PUT∆BAN ANY∆CHAR∆VEC   ⍝ ... update banner details

)OFF              ⍝ ....  go and try it!


    FI PUT∆BAN BANNER
    -----------------

[1]  ⍝ PATCH IN BANNER TO FILE FI .....

[2]   FI ⎕NTIE ¯1
[3]   (631↑BANNER,⎕TCNL,630⍴⎕TCNUL)⎕NREPLACE ¯1,BNR
[4]   (6⍴⎕TCNUL)⎕NREPLACE ¯1,SER
[5]   (3⍴⎕TCNUL)⎕NREPLACE ¯1,VER
[6]   ⎕NUNTIE ¯1
    ∇
```

Technical Correspondence

Yet More Benchmarks ...

From N J Small 16th September 1987

Sir,

In case it has not been drawn to your attention before, you may be interested in the following comparison of timings made using APL*PLUS/PC (running on an IBM PC/XT286) with those from "APL: A Design Handbook for Commercial Systems" (page 136). (I was unable to locate a reference to the machine and system to which these timings relate.) <<IBM 3032 running VS APL under VSPC .... Ed>>

```
   Command                             APL*PLUS/PC     ???
M←100 500ρ'RABBITS '              :        .49 s
M1←M[⍳3;]                         :        .16 s     1.4  ms
M1←((1↑ρM)↑3ρ1)⌿M                 :       1.75 s     0.69 ms
M1←(3,1↓ρM)↑M                     :       1.92 s     0.29 ms
```

Of course, the ratios of the various methods vary quite a bit with the size and shape of the object being handled (times approximate):

```
M←100 300ρ'RABBITS '              :        .27 s
M1←M[⍳3;]                         :        .06 s
M1←((1↑ρM)↑3ρ1)⌿M                 :       1.10 s
M1←(3,1↓ρM)↑M                     :       1.15 s

M←10 300ρ'RABBITS '               :        .06 s
M1←M[⍳3;]                         :        .06 s
M1←((1↑ρM)↑3ρ1)⌿M                 :        .16 s
M1←(3,1↓ρM)↑M                     :        .16 s

M←100 30ρ'RABBITS '               :        .05 s
M1←M[⍳3;]                         :        .05 s
M1←((1↑ρM)↑3ρ1)⌿M                 :        .16 s
M1←(3,1↓ρM)↑M                     :        .11 s
```

I was stimulated to perform these timings after observing that, if one is dealing with vector data, from a file, it can take much longer to take characters from the vector than it took to read the vector in the first place; in fact it can be much faster to re-read sub-strings than to pull them out of the vector in the workspace using indexing, if they are in a buffer. (Using take is worse than a read from the disk, for the case illustrated.) My figures were:

```
Z←⎕NREAD ¯1 82 35497    0                 :      .38 s
ZZ←⎕NREAD ¯1 82   999    0                :      .11 s
ZZ←⎕NREAD ¯1 82   999  999                :      .00 s
ZZ←⎕NREAD ¯1 82   999 9999                :      .11 s
ZZ←999↑Z                                  :     2.97 s
ZZ←9999↑Z                                 :     3.02 s
ZZ←Z[⍳999]                                :      .11 s
ZZ←999⍴Z                                  :      .00 s
ZZ←((⍴Z)↑999⍴1)/Z                         :     6.37 s
ZZ←((⍴Z)↑999⍴1)                           :     3.24 s
```

Further investigation revealed the following, which I suppose one might have guessed at the start, if one had thought hard enough: certain magical properties of a well-known magical number combine to cause enormous overheads when the interpreter switches from two-byte to eight-byte integer arithmetic for indexing.

```
ZZ←(¯1+2*15)↑Z                            :     3.24 s
ZZZ←999↑ZZ                                :      .05 s
ZZ←(2*15)↑Z                               :     3.24 s
ZZZ←999↑ZZ                                :     2.74 s
```

I hope that the above provides some entertainment, even if it should be of no greater value to you.

Yours sincerely,

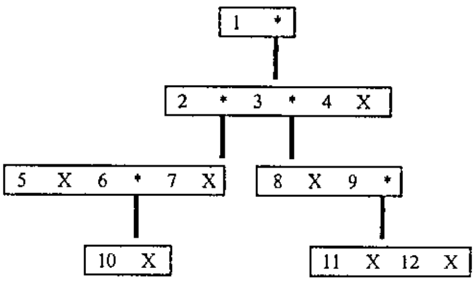
Nicholas Small,
I.T. Systems Support,
Room 236,
National Westminster Bank,
41 Lothbury,
LONDON EC2P 2BP.

Mail Point 188,
IBM UK Laboratories,
Hursley Park,
WINCHESTER,
Hants. SO21 2JN

Sir,

It was interesting to read in VECTOR 4.1 of Anne Wilson's delight in finding how adaptable APL is to handling trees. Perhaps it would delight her further to see how the problems she discusses might be tackled in APL2.

First, model your tree! One way is as a vector with an even number of items - odd items are keys, even items are trees or the enclosed empty vector. In order to avoid a plethora of quotes, keys will be numbers rather than letters, so that the "typical tree" is :



Its APL2 definition is :

```
        X←⊂⍳0
        TREE←1(2(5 X 6(10 X)7 X)3(8 X 9(11 X 12 X))4 X)
```

To solve the subtree size problem use a function PATH which is the inverse of "pick" :

```
[0]     Z←L PATH R;T
[1]     Z←⍳0
[2]     →(0=≡R)/0
[3]     T←(L∊¨∊¨R)⍳1
[4]     Z←T,L PATH T⊃R

        7 PATH TREE
2 2 5
        8 9 10 PATH¨⊂TREE
 2 4 1  2 4 3  2 2 4 1
```

Subtrees and subtree size are then given by the functions

```
[0]    Z←L SUBT R
[1]    Z←(L STPATH R)⊃R

[0]    Z←L STPATH R;T
[1]    Z←(⌽(⍴T)↑1)+T←L PATH R
```

SUBT returns the subtree associated with key L :

```
       2 SUBT TREE
 5     6   10        7
```

and the sizes of subtrees are given by

```
[0]    Z←L SIZE R
[1]    Z←⍴∊(L STPATH R)⊃R

       (⍳12)SIZE¨⊂TREE
 11  4  4  0  0  1  0  0  2  0  0  0
```

STPATH is also used in functions to cut, copy, and cut and paste subtrees :

```
[0]    Z←L CUTFROM R
[1]    Z←R
[2]    ((L STPATH Z)⊃Z)←⊂⍳0

       2 CUTFROM TREE
 1   2    3   8    9   11    12        4

[0]    Z←L MOVE R
[1]    Z←R
[2]    ((L[1] STPATH Z)⊃Z)←(L[2] STPATH Z)⊃Z
[3]    Z←L[2] CUTFROM Z
```

e.g. cut the subtree at node 3 and paste it to node 7 :

```
       7 3 MOVE TREE
 1   2   5    6   10      7   8    9   11    12        3    4
```

Ancestors are found by a function very similar to PATH :

```
[0]    Z←L ANCIN R;T
[1]    Z←⍳0
[2]    →(L∊R)/0
[3]    T←(L∊¨∊¨R)⍳1
[4]    Z←R[T-1],L ANCIN T⊃R

       8 ANCIN TREE
1 3
```

All sets of ancestors are given by :

```
          (⍳12)ANCIN¨⊂TREE
   1  1  1  1 2  1 2  1 2  1 3  1 3  1 2 6  1 3 9
 1 3 9
```

Finally to obtain nodes at the various levels, an auxiliary function and an auliliary operator are defined. The function ODDS selects the odd items from a vector :

```
[0]    Z←ODDS R
[1]    Z←((⍴R)⍴1 0)/R
```

and the operator LEV penetrates the depth of R to a prescribed level G before F is applied :

```
[0]    Z←(F LEV G)R
[1]   →(G<≡R)/L1
[2]   →0,⍴Z←F R
[3]   L1:Z←F LEV G¨R

          ODDS LEV 5 TREE
  1  2 3 4
```

(Note that the depth of the tree is 6).

The function LAYEROF completes the picture, and as with SIZE, all layers can be computed simultaneously using "each".

```
[0]    Z←L LAYEROF R
[1]    Z←(∊ODDS LEV L R)~(L≠≡R)/∊ODDS LEV(L+1)R

          4 LAYEROF TREE
5 6 7 8 9
          6 5 4 3 LAYEROF¨⊂TREE
  1  2 3 4  5 6 7 8 9  10 11 12
```

Not only do the APL2 techniques appeal (to me at least!), but the original 7 pages have been reduced to 3 - a 57% reduction factor in favour of APL2!

Yours,

Norman Thomson.

Competition Result - Sweeten your Combinations

Derek Wilson

The problem was to write a function COLOURS which returns the probability of having exactly 1, 2, 3, ... N different colours in a pack of sweets. There are N colours available, equally probable, and the pack is filled randomly with M sweets. For example:

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

The competition attracted thirteen entries from Canada, Australia, USA and the UK, using VS APL, APL*PLUS PC and IBM PC APL, and the approaches used to tackle the problem can be categorized roughly in four ways.

1. By Induction / Recursion

The problem can be solved by induction on the number of sweets, M, because if there are R distinct sweets in a pack of size M, then either

a) There were R distinct sweets in the first (M-1) sweets and the next sweet is the same as one already chosen, or

b) There were (R-1) distinct sweets in the first (M-1) and the next one is a new colour.

The initial conditions are easy to set up and this approach gives rise to recursive or iterative solutions.

Alternatively, the recursion can be done on the number of colours present using the following logic provided by Zeke Hoskin:

There are N*M ways to pick M sweets of N colours but

((N-1)*M) x (N-1)!N of these will show fewer than N colours, but

((N-2)*M) x (N-2)!N of those will show fewer than (N-1) colours etc. etc.

Several of the solutions followed this kind of theme.

2. Exact Mathematical Expression

Various entrants built on the logic described above to arrive at an exact mathematical expression for the answer, and some found the answer in Statistics textbooks (Feller was the favourite).

One expression is :

```
(N*-M)×(r!N)×⌹(r*M)-.×r∘.!r←⍳N     ⍝ ⎕IO=1
```

3. Counting the Options

This involves calculating all the different option and counting up those which meet the criteria. Various entrants devised neat means of doing this using local functions and matrix division to speed up the calculations and reduce the sizes of the intermediate arrays.

4. Simulation

Richard Connor belied his statement that "I recognize the Candy packing machine is a stochastic process, but my knowledge of them is confined to spelling 'stochastic'" by submitting two simulation functions and asking the judges to spin a coin to decide between them!

Not surprisingly those solutions in the first two categories were the most efficient, but I don't dismiss the simulation or counting approaches too readily. Faced with a practical one-off problem of this type the Marketing Manager is interested in the speed of obtaining an answer which is correct to 3 or 4 decimal places; not that it was achieved using an APL function which took 134 milliseconds to run and and which was index origin independent! If the hypotheses of randomness or that the colours are equally likely are altered in any way then a theoretical approach is well nigh impossible.

By testing the entries with different values of N and M it was fairly easy to arrive at the three best solutions without having to resort to pathological input data, although it was interesting to see that some of the non-looping solutions produced nonsense if a value of N much greater than M was used. Those entrants who went to a lot of trouble to remove loops or recursion will be annoyed to learn that the iterative solutions are much quicker for larger values of M.

Prizes

Joint first prize goes to Prof. Alan Hawkes of Swansea and Dr. Roger Glassey of Berkeley whose solutions are both efficient and very readable, and third prize to John Buckland of Camberley whose answer uses different algorithms depending on the size of the input data.

Thanks to them and to all the other entrants.

```
        Z←N COLOURS C;R
        ---------------
    [1]      ⍝ RECURSIVE SOLUTION BY ALAN HAWKES TO COLOURS PROBLEM
    [2]      ⍝ PROBLEM FROM VECTOR VOL3 NO4 (APRIL 1987), P 114
    [3]      ⍝ RESULT IS VECTOR OF PROBABILITIES OF 1 TO C DIFFERENT COLOURS
    [4]      ⍝ IN N INDEPENDENT SELECTIONS FROM C EQUALLY LIKELY COLOURS.
    [5]      ⍝ N AND C ARE POSITIVE INTEGER SCALARS.
    [6]       →(N=1)/INIT
    [7]       R←(1-⎕IO)+⍳C
    [8]       Z←((R×Z)+¯1↓0,(C-R)×Z←(N-1)COLOURS C)÷C
    [9]       →0
    [10] INIT:Z←C↑1
        ∇

                P←S COLOURS C;I;K
                -----------------

            [1]      ⍝ P[K] is prob that K colours of a possible C will be found
            [2]      ⍝ in a pack of S sweets
            [3]      ⍝ Solves a difference equation to calculate prob vector P.
            [4]      ⍝ At iteration I, P[K] is probability for I sweets
            [5]      ⍝ Adding 1 more sweet to the box adds a new colour with
            [6]      ⍝ prob (C-K)÷C, but a matching colour with prob K÷C.
            [7]      ⍝ If S=0, return the 0 vector, else P[1]←1
            [8]
            [9]
            [10]      P←(S>0),1↓C⍴0 ⋄ I←0
            [11]      K←(1-⎕IO)+⍳C
            [12] LOOP:→(S≤I←I+1)↑0
            [13]      P←((P×K)+(0,¯1↓P)×1+C-K)÷C
            [14]      →LOOP
            [15]
            [16]     ⍝ Roger Glassey
            [17]     ⍝ Dept of IE&OR
            [18]     ⍝ University of California Berkeley
            [19]     ⍝ 10 June 1987
                ∇

    R←S COLOURS N;I;W;X;Y;Z
    -----------------------

[1]    ⍝R IS PROB OF 1-N COLOURS APPEARING IN PACKSIZE S - FOR SMALL W/S
[2]     →(~⍴R←⍳^/2=(⍴I),+/×I,-1|I←S,N)/0 ⋄ →(20<+/I)/L1 ⍝ POS INTS ONLY
[3]     W←¯1↓Y←!X←0,⍳N ⍝ FOR SMALL S AND N
[4]     R←((¯1↑Y)÷⌽W)×-⌿(X∘.≤X)×(-X←¯1↓X)⌽⍉(((1↓X÷N)*S)÷1↓Y)∘.÷W ⋄ →E
[5]  L1:→(S=1↑R←N↑I←1)/0 ⋄ X←X,[1.5]⌽X←(⍳N)÷N ⍝FOR LARGE S AND N
[6]  L2:R←+/X×R,[1.5]0,¯1↓R ⋄ →(S>I←I+1)/L2
[7]   E:R←,'F14.11' ⎕FMT R
[8]    ⍝ JCL Buckland
     ∇
```

A single source for APL

Advice

Design

Application Development

Education

APL software

Support

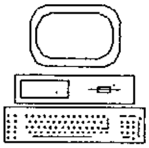
COCKING & DRURY LTD.

THE APL PROFESSIONALS

180 Tottenham Court Road,
London W1P 9LE
Tel: 01 – 436 9481

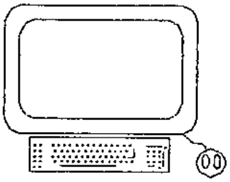
A single source for APL*PLUS

APL*PLUS PC



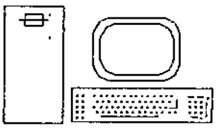
for PCs and the PS2

APL*PLUS UNX



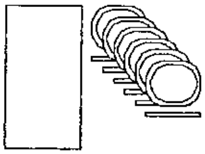
for UNIX systems

APL*PLUS PRO



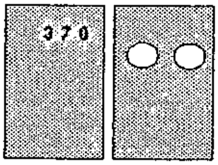
for the PS2 model 80

APL*PLUS VMS



for the VAX range

APL*PLUS VM & MVS



for IBM mainframes

COCKING & DRURY LTD.

THE APL PROFESSIONALS

180 Tottenham Court Road,
London W1P 9LE
Tel: 01 - 436 9481

IBM Personal Computer APL/PC Version 2.0

6391329

IBM Personal Computer APL/PC Version 2.0, is a low cost, full function APL interpreter with a high degree of VS APL compatibility. It contains a wealth of auxiliary processors for a wide range of functions and interfaces to external devices.

- Emulates 8087 or 80287 if the co-processor is not present
- RS232 support
- IEEE-488/GPIB support
- Co-operative processing via IBM 3278/9 adapter
- Interface to IBM Macro Assembler and Professional Fortran
- APL2 GRAPHPAK compatible workspace provided
- Can run DOS functions and applications under APL
- Cover Workspaces for auxiliary processors

The interpreter, workspaces and auxiliary processors are supplied on three double-sided diskettes packaged with a comprehensive manual, quick reference card and a keyboard template. The manual includes setup, installation, tutorial and reference sections.

A separate package is supplied, containing a replacement ROM for the IBM Monochrome or Colour display adapters and a ROM puller. A program to load the APL font into the IBM Enhanced Graphics Adaptor is also included.

- Available from Authorised IBM PC Dealers.

IBM (UK) International Products Ltd
West Cross House
2 West Cross Way
Brentford
Middlesex TW8 9DY

A Note on Weighted Least Squares

by Peter Kelly and Norman Thomson

It is well known that the APL domino function includes a routine to obtain least squares fits of a straight line to a set of observations. Specifically, suppose that the model to be fitted to a set of multivariate data containing n observations is

$$y = b_0 + b_1x_1 + b_2x_2 + \dots b_px_p + e$$

where the residuals e_i ($i = 1,2,\dots n$) are assumed to be identically and independently distributed with an error variance which has to be estimated along with the b's. Suppose further that the data is presented in the form of a matrix with n rows and (p+2) columns, the leftmost of which gives the y values, and the remainder the x-values, starting with a column of 1's to correspond to the fitting of the constant term b_0. The vector of regression coefficients is then given by :

```
        BS : R[;1]⌹0 1↓R
```

and the residuals by

```
        ES : R+.×¯1,BS R
```

The error variance is given by

```
        EVAR : (+/(ES R)*2)÷1+-/⍴R
```

and the variances of the b's by

```
        BVAR : 1 1⍉(EVAR R)×IMS R
```

where IMS stands for "Inverse Matrix Squared", a function which as $(X'X)^{-1}$ occurs repeatedly in analysing residuals. It is defined as

```
        IMS : ⌹(⍉X)+.×X←0 1↓R
```

In practice there are situations in which the assumptions that the residuals are identically distributed is not tenable, e.g. if the population is in fact several subpopulations whose residual variances are proportional to one of the x values, or if the residual variances corresponding to particular x values are known as the result of repeated experimentation, as might happen say in a physics experiment. The appropriate analysis is then weighted least squares. Assume that the variance corresponding to the ith. observation is w_i times the error variance, where (w_1, w_2,... w_n) are the items of a known vector W. The appropriate regression coefficients, error variance, and variances of the b's are the BS, EVAR, and BVAR of the data matrix

```
        (DIAG ÷W*.5)+.×R
```

where DIAG is a function which constructs a diagonal matrix with the weights $w_1 \dots w_n$ as the leading diagonal, e.g.

```
        DIAG : ((2⍴⍴R)⍴R)×T∘.=T←⍳⍴R
```

To give a specific example, consider the data

x : 3	5	8	12	15	17	20
y : 0.8	1.2	2.5	4.9	8.3	12.3	17.6

as a candidate for a least squares fit

$$y = b_0 + b_1x_1 + e$$

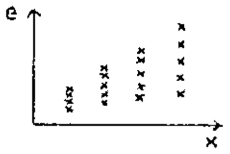
so that R is

```
 0.8   1   3
 1.2   1   5
 2.5   1   8
 4.9   1  12
 8.3   1  15
12.3   1  17
17.6   1  20
```

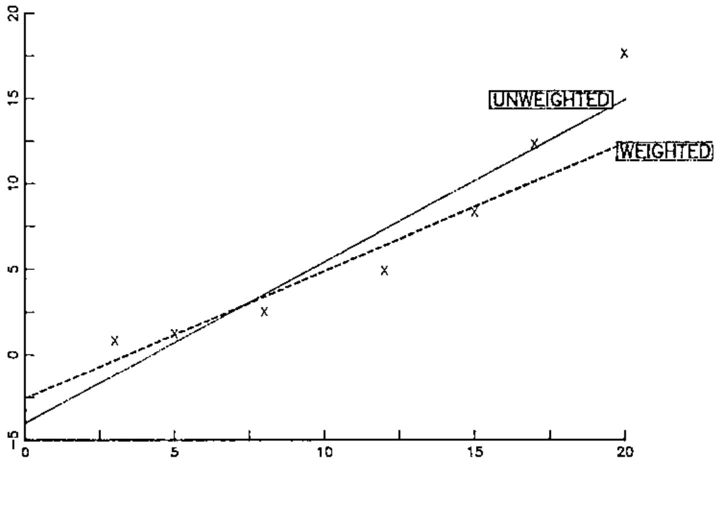
The coefficients b_0 and b_1 are given by

```
      BS R
¯4.018 0.9466
```

If, however, there is evidence that the spread of the residuals increases with x so that a plot of e vs x has the appearance



then weighted least squares is appropriate. Assuming weights proportional to x gives a regression line y = -2.562 + 0.8192x. As the graph below shows the effect is to downgrade the importance of the last two points which are in the relatively more variable x region.



The more the weights are "strengthened", e.g. by taking them to be the squares of the x values which gives $y = -1.721 + 0.709x$, the more closely does the weighted squares line track the smaller value points.

If the model

$$y = b_0 + b_1x_1 + b_2x_2 + e$$

is chosen, i.e. the data matrix is changed to

```
      R,R[;3]*2
```

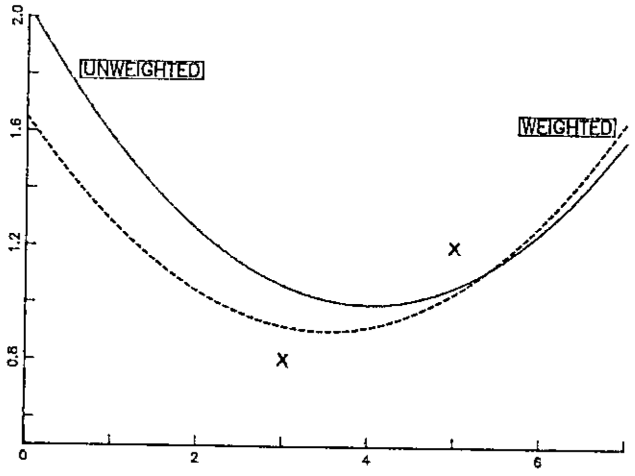
the fitted quadratics are

$$y = 2.037 - 0.5199x + 0.06482x^2 \text{ (unweighted)}$$

and

$$y = 1.649 - 0.4258x + 0.06066x^2 \text{ (weighted)}$$

The effect of weighted least squares is to cause the regression parabola to approach closer to the lowest x-value point as the graph below illustrates.



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.

APL*PLUS/PC and Assembler (Part 2)

by P Wortham (UNIWARE)

Introduction: How APL Manages Memory

In a previous paper (VECTOR 4.2), we have run an Assembler-written program from APL. We are going to see how we can have access to APL variables.

The ⎕STPTR function has been evoked in the previous paper. To understand what its use may be, we first have to know how APL manages its memory.

An APL variable is stored in memory in consecutive words. Ahead of the data words, we can see the variable's descriptor, i.e. its type (character, integer or real), its rank and dimensions. Some more information is present for APL's use. Considering the vector:

```
        A←⍳5
```

In memory, we can find the following bytes, where X represents a byte that does not have a useful meaning for us:

```
2  0  X  X  X  X  2  1  5  0  1  0  2  0  3  0  4  0  5  0  255...
----                  ----  ----  ---------------------------  ------
 a                     b     c                 d                  e
```

A variable is always stored in a multiple of 16 number of bytes. If positions are not all used, the filling value is 255. The vector X uses 20 bytes, plus 12 filling bytes in the (e) area. Its size is 32 bytes.

The (a) area is equal to the total number of bytes used by A, divided by 16, that is 2 in this case. This area is integer coded, so we have 2 (low half) before 0 (high half).

The (b) area represents the variable's type: 1 for character, 2 for integer and 8 for real, followed by the rank (1 for this vector).

If the variable is a scalar, its value comes just after the type: in this case, there is no dimension description (c area). In the other cases, we find all dimensions, in the same order as ⍴A. In this example, the variable is a vector, its dimension is 5, so we have 5 and 0. If the variable has one dimension that exeeds 32767 elements, the byte holding the type is 16 plus the normal value, i.e. 17, 19 or 24. Then the dimensions are 2 words long.

The (d) area holds the data bytes, in the order given by ravel A.

Now we now know how a variable is stored, but we still have to know its location in memory!

When APL needs memory, for a variable or function creation, or for an intermediate result, it searches for free memory in high addresses. If the free space is not sufficient, then it reoganizes all the memory, i.e. it suppresses all the erased variables and functions (This is called a garbage collection'). Then it tries again. If the error occurs once more, we receive a WS FULL message.

This process means that a variable address is not steady. The only steady information we can get is ⎕STPTR's result. A variable's address is given by:

```
256 ⊥ ⎕PEEK 1 0 + ⎕STPTR  'VARIABLE'
```

The result is exact for a few moments, but it may be wrong before when can use it!

So, the parameters we will give to an assembler written routine will not be real addresses, but ⎕STPTR results. This will be sufficient to find the real addresses in the routine, and the APL objects will not move during the Assembler routine execution, while APL is pending. We can calculate an address in Assembler with:

```
MOV  ES,[BX] ; if BX = ⎕STPTR 'VARIABLE', then ES is the address.
```

This instruction will work if the DS register has the value corresponding to a ⎕SEQ of iota zero. This is the value which is always passed by APL to DS. If you change its value, you have to give the old value before any address search. If you do not remember that, you will have wrong results, or your computer may hang.

Here is the listing I promised you:

```
CODE     SEGMENT BYTE
         ASSUME  CS:CODE
; AX = STPTR BINARY VECTOR
; result = ax  =      1=+/binary
BOOL1    PROC    FAR
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         NOP
         MOV     BX,AX              ; BX = STPTR A
         MOV     ES,[BX]            ; ES = ADRS A
         MOV     AX,ES:WORD PTR 6   ; A'S RANK AND TYPE
         CMP     AL,2               ; INTEGER ?
         JNE     ERR                ; ERROR IF NOT
         MOV     DI,10              ; POINTS TO 1RST VALUE
; ES:DI POINTS TO THE BEGINNING OF THE VECTOR
         MOV     CX,ES:WORD PTR 8   ; DIMENSION IF IT'S A VECTOR
         SUB     BX,BX              ; NUMBER OF 1, INITIALIZED TO 0
         CMP     AH,1               ; TAKE A LOOK TO THE RANK
         JE      BOU                ; GO AHEAD IF IT'S A VECTOR
         JL      OK                 ; OK IF IT IS A SCALAR
ERR:     INT     0                  ; "DOMAIN ERROR" IF RANK > 1
OK:      MOV     CX,1               ; SCALAR, SO 1 ELEMENT
         DEC     DI                 ; POINTER TO THE SCALAR'S VALUE
         DEC     DI
; LOOP ON EACH VECTOR'S ELEMENT
LOOP:    JCXZ    DONE               ; DONE IF IT IS THE LAST ELEMENT
         MOV     AX,1               ; 1 IS THE VALUE WE ARE LOOKING FOR
         REPNE   SCASW              ; GO TO THE NEXT 1
         JNE     DONE               ; END OF VECTOR, THERE IS NO OTHER 1
         INC     BX                 ; WE HAVE SEEN A 1
         REPNE   SCASW              ; GO TO NEXT 1
         JNE     DONE               ; DONE IF NO OTHER 1
         INC     BX                 ; IF WE FIND 1
DONE:    XOR     AX,AX              ; INITIALIZE THE RESULT
         CMP     BX,1               ; THE NUMBER OF 1
         JNE     DDONE              ; IF WE DO NOT HAVE EXACTLY ONE 1
         INC     AX                 ; IF WE HAVE ONE 1
DDONE:   RET                        ; AND RETURN TO APL
BOOL1    ENDP
CODE     ENDS
         END
```

The routine has to be assembled, linked and translated to binary. We can get the result from APL with:

```
   'BOOL1.BIN'⎕NTIE ¯1
   CODE←10↓⎕NREAD ¯1 82,⎕NSIZE ¯1
```

We need to drop the first 10 elements which correspond to the ten NOP instructions we find in the beginning of the Assembler code. This operation is not always necessary, but it is a good habit to have.

The code is stored, in character format, in APL variable CODE. The same code, in integer format, may be obtained by:

```
    CODE←163 ⎕DR CODE,(2|⍴CODE)⍴⎕AV
```

Writing the APL function is now an easy operation. We can test this function with vectors of different lengths in order to know if this function is faster or slower than the equivalent APL expression. We can see that for little objects, the APL solution is faster, but for a large vector, the Assembler routine is much faster. For instance:

```
          V←10000↑1
          1=+/V
    1
          BOOL1 V
    1
```

For such a variable, the Assembler routine is 12 times faster than its APL equivalent. But we have to remember that we should use the APL expression for arguments of more than 32767 elements because the Assembler routine will give an error in that case. It is possible to write an Assembler routine that handles large vectors. Of course, such a routine will be more complex to write, and the execution time will be a bit slower for little objects. Another point is that the APL expression is valid for objects with a rank larger than 1, and for objects containing not only 0's and 1's.

We now know how to read an APL variable. We can update a variable as easily. We also have the possibility to create objects directly from Assembler.

Conclusion

This little function shows what we can expect from Assembler: when we want to speed up a very specific operation, the code we have to write is very short, and we can considerably reduce the computing time. Often, we will have to reduce the domain of the operation, and we will avoid unecessary tests. In our example the domain is reduced to vectors of less than 32768 elements containing only 0's and 1's.

British APL Association Software Library

Software Submission (1)

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. Please Use BLOCK CAPITALS for all items except number 18.

0. Submission date:__________

1. Name of donor:______________________ 2. Daytime phone(*):________

3. FULL address(*):____

____

4. Electronic mail addresses(*):________________________________

5. Short disk title:____

6. Brief description of disk :__________________________________

____

____

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 Software Library

Software Submission (2)

16. File names and descriptions. This information will be made publicall available in the software library catalogue. Please save us some wor 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. Use any extra space above for comments you wish to appear in the software 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 Software Library
                         c/o Dave Ziemann
                         Flat 3, 63 Queens Crescent
                         London NW5 4ES, ENGLAND
```

British APL Association Software Library

Order Form

To: Dave Ziemann
Flat 3, 63 Queens Crescent
London NW5 4ES, ENGLAND

PLEASE SUPPLY THE DISKS CIRCLED BELOW:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99	100							

TOTAL NUMBER OF DISKS ORDERED (please double-check): ______

at 2 Pounds per disk :	______	BAA Member's rate
OR at 3 Pounds per disk :	______	Non-member's rate
I want to join the BAA, and enclose the 10 Pounds annual membership fee :	______	
Postage and handling :	1.00	
Add 2 Pounds for orders outside the UK:	______	
Total order - remittance included:	======	

Your signature: ________________________________

Your name: ________________________________

Full address: ____

____

____

Please make cheques payable to the British APL Association.

Payment can be made in US Dollars - pay 1.5 US Dollars for each Pound.

All orders must include payment - allow 30 days for delivery.

BAA membership year runs from 1 May - 30 April. People joining part way through the year will receive appropriate back numbers of Vector.

Software is accepted by the British APL Association on good faith and we do not vouch for or make any claims regarding donated software. The BAA shall not be held responsible for damage caused by the use of library software, and shall not be liable in any way in the event that submitted material proves to be the subject of third party copyright. Your signature above accordingly indemnifies the BAA from all liability.

Index to Advertisers

APL People (half page)	4
APL Software Ltd (half page)	122
Cocking and Drury	16,17,112,113
Dyadic Systems Ltd	2
HMW Programming Consultants	72
IBM PC	114
IBM Sweden	118
MicroAPL	6
VECTOR back numbers	19
APL Booklist	20

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: 0707-325161

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):	87/88	
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.

Typeset using MS Word 3.0 on Xerox 3700

Printed in England by Short-Run Press Ltd, Exeter