MAY/JUNE 1981
Volume 9/Number 3

I.P. Sharp newsletter

CONTENTS

NPA

U.S. NATIONAL PLANNING ASSOCIATION ECONOMIC DATA BASE

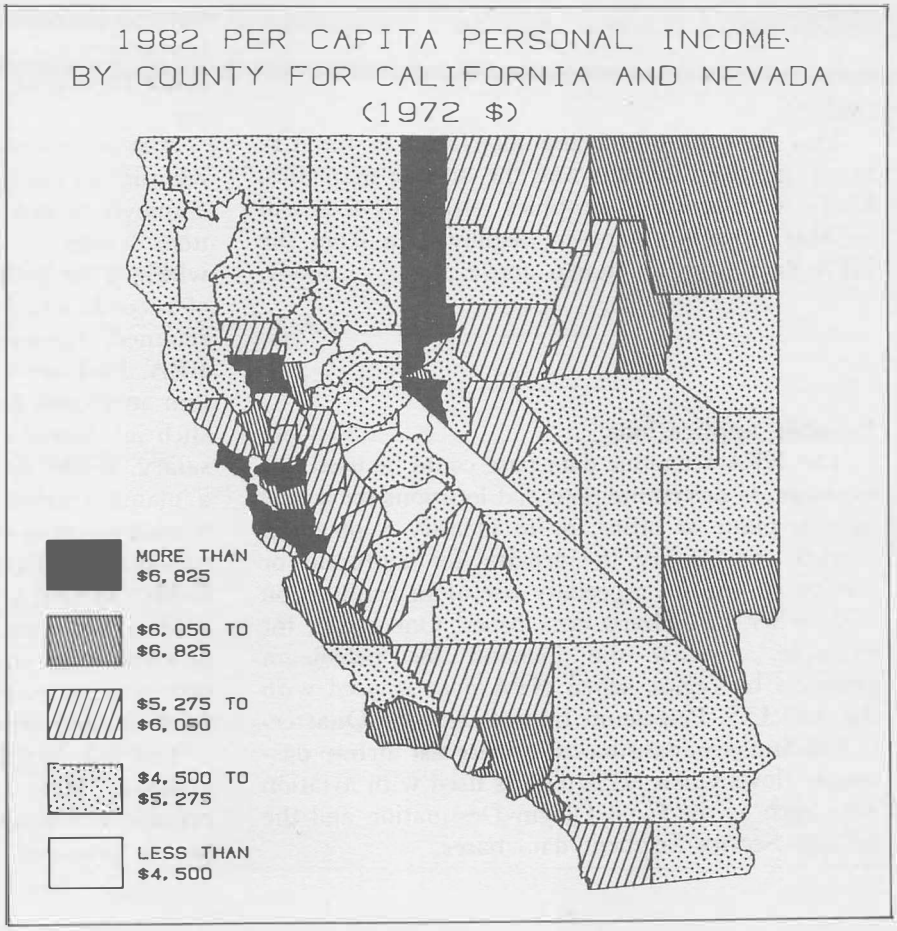
Bill Flowers, Toronto

In response to the increasing demand for economic data, the NPA economic data base has been made available to all users of the I.P. Sharp system, with no surcharge or subscription fee. I.P. Sharp Associates is currently the only on-line supplier of this data.

The NPA data base contains over 200,000 yearly time series relating to the U.S. economy, historically from 1967 and projected to the year 2000. The data is aggregated by county, state, region, BEA (Bureau of Economic Analysis) economic area, SMSA (Standard Metropolitan Statistical Area), and the total U.S. In all there are over 3,600 areas.

For each area there are 56 time series covering employment, income and population. The data base is updated yearly with data received from the National Planning Association in Washington, D.C.

Access the NPA data base via 39 *MAGIC*: type *NPASET*. On-line information is available by typing *DESCRIBENPA*; a highspeed listing of the area codes can be requested by typing
NPAECO 'DIRECTORY'.



The data can be retrieved by using the access command *NPAECO*, which has the syntax:

'area code(s)' *NPAECO* **'account code(s)'**

or

'area code(s)' *NPAECO* **account number(s)**

where multiple area codes and account codes are separated by commas. For example, to display projected earnings per employee, by industry,

```
CLEAR ◊ TIMESERIES ◊ YEARLY,DATED 80 TO 84 ◊ NOAUTOLABEL
NEWTITLE 'EARNINGS PER EMPLOYEE BY INDUSTRY TYPE FOR U.S.A.'
TITLE '(1972 $)'
SCALE 1
T←'USA' NPAECO 'YEMIN,YECON,YEMFGN,YEMFGD,YETCPU'
T←T AND 'USA' NPAECO 'YETRDW,YETRDR,YEFIRE,YESVS'
PUT T DIVIDED BY 'USA' NPAECO 10 11 13 14 15 16 17 18 19
NEWLABEL 'MINING, CONSTRUCTION,MANUFACTURING:⍺ DURABLE, NON-DURABLE'
LABEL 'TRANSPORTATION⍺ COMMUNICATIONS⍺ AND PUBLIC UTILITIES'
LABEL 'TRADE:⍺ WHOLESALE, RETAIL,FINANCE INSURANCE⍺ AND REAL ESTATE'
LABEL 'SERVICES'
'H' DISPLAY ABOVE
```

EARNINGS PER EMPLOYEE BY INDUSTRY TYPE FOR U.S.A.
(1972 $)

	1980	1981	1982	1983	1984
MINING	15,736	15,843	16,157	16,558	16,945
CONSTRUCTION	12,499	12,488	12,638	12,852	13,052
MANUFACTURING:					
DURABLE	9,904	9,895	10,014	10,183	10,341
NON-DURABLE	12,246	12,238	12,387	12,600	12,749
TRANSPORTATION					
COMMUNICATIONS					
AND PUBLIC UTILITIES	13,809	13,800	13,968	14,208	14,431
TRADE:					
WHOLESALE	11,625	11,612	11,749	11,944	12,128
RETAIL	6,055	6,052	6,126	6,233	6,332
FINANCE INSURANCE					
AND REAL ESTATE	10,689	10,896	11,065	11,392	11,631
SERVICES	8,248	8,242	8,342	8,486	8,619

This data base is being interfaced to the I.P. Sharp graphics utilities and the on-line map files. Users with access to graphics output devices will be able to draw maps to display data from the NPA data base, as shown above.

Possible applications

The NPA economic data base could be used, for example, to determine potential locations for a new branch plant or office, or to identify a potential market and forecast the demand for a product or service. It can be especially useful in conjunction with other I.P. Sharp data bases. One could, for example, forecast the demand for petroleum products by region when NPA data is used with the API U.S. Petroleum Imports and the Quarterly Oil Statistics data bases, or forecast airline passenger flows when NPA data is used with aviation data such as the CAB Origin-Destination and the ER586 Service Segment data bases.

MABRA

A NEW RELEASE

Hugh Hyndman, Toronto, and
Steve Halasz, Rochester

By far the majority of data processing applications are record administration systems of one kind or another. MABRA satisfies the demands of SHARP APL customers for a flexible, easy-to-use, generalized record administration system which is able to cope quickly with organizational changes. Originally developed to support a personnel administration system for BL Cars (British Leyland) covering that company's 100,000 employees, MABRA is fast becoming I.P. Sharp's most widely used package.

MABRA is used internally at I. P. Sharp for many management information systems, and by SHARP APL customers in hundreds of applications in personnel administration, employee benefits administration, manufacturing, transportation, order processing, marketing, science and engineering.

Almost any system made up of a collection of records can easily be managed using MABRA. An employee record within a personnel system, a stock item within a warehouse system, or a product within a manufacturing system, are all examples of **records** which can efficiently and effectively be retained, retrieved, and maintained using MABRA. Each record in such an application has certain attributes. Each employee record could include such information as the employee's name, address, salary, grade, or department. On the other hand, a manufacturing system might record such information as serial number, unit cost, and retail destination. In MABRA, these attributes are called **fields**. MABRA lets you add, modify, and delete fields at any time. You can select records according to a value or range of values in any field, and full provision is made for the addition, modification, deletion, and reporting of records.

You use MABRA through a simple prompted dialogue. Help is available on request at each prompt, and simple and thorough user documentation is provided. You do not have to know any-

thing about computers or computer programming in order to set up, maintain, and utilize a MABRA system.

MABRA is continually evolving in reponse to the needs of our users, and a number of major enhancements to MABRA are now available. New commands enable users to

- **link** MABRA systems;
- set up **pseudo fields** based on calculations on real fields; and
- generate a wide range of standard **reports**.

Linked Systems

A single application sometimes breaks down naturally into two related MABRA systems. For example, suppose that a personnel system is maintained which contains fields for employee name, address, grade, and social security number, and that a related pension administration system is maintained which includes a field for social security number, and fields for employer and employee contributions to the plan. Then we can view the social security number, which occurs in both MABRA systems, as a **key field** which can be used to **link** the systems, as shown in the following example:

```
:ENTER◊10
* DATA MODE, SYSTEM IS 10: PERSONNEL SYSTEM
10:LS
* LINK SYSTEMS
* CREATOR AND SYSTEM NUMBER: 20
* TOTALLING REQUIRED? (Y/N): N
* KEY FIELD ON CURRENT SYSTEM: SSNUMBER
* KEY FIELD ON LINKED SYSTEM: SSNUMBER
* LINKED SYSTEM RECORD CONSTRAINT:  ALL
     1215 FOUND.
* FIELDS TO LINK:  EMPEECONTR,EMPORCONTR
* NEW FIELD NAMES:  EMPEECONTR,EMPORCONTR
* OK TO LINK? (Y/N):  Y
* SYSTEMS LINKED.
```

In this example, *EMPEECONTR* and *EMPORCONTR* are the field names for employee and employer contributions respectively in the pension system, system 20. By the procedure described above, these fields are linked with the records in the personnel system, system 10, by matching records which have the same social insurance number *SSNUMBER*. The newly linked fields are given the same names as they had in the pension system, but you could specify completely different names. The newly linked fields can now be used as search, print, or sort fields. A session to list these fields together with corresponding records in the personnel system can proceed as follows:

```
10:LR
* LIST RECORDS
* CONSTRAINTS: ALL
     50 FOUND
* PRINT FIELDS:  NAME,GRADE,SSNUMBER,EMPEECONTR,EMPORCONTR
* SORT FIELDS:  GRADE,NAME
* ALIGN PAPER, TYPE SPACE, CR TO START.

NAME              GRADE      SSNUMBER  EMPEECONTR  EMPORCONTR

BAKER,MARTIN P.      20   384-38-3847     2344.23     4232.23
DALMOTH,JANE R.      20   837-32-3442     4221.11     3234.15
LINGHORN,ELLIOT      20   423-23-3556     6433.36     7435.31
ANDERSEN,MARGARET    21   873-35-9337     7325.64     6436.97
BREWER,PAMELA        21   534-35-8342     6352.25     7342.12
******
```

In the example described above, there is one record in the linked (pension) system for each record in the base (personnel) system. MABRA also handles the cases where many records in the linked system correspond to one in the base system, and where many records in the base system correspond to one in the linked system.

A sales order system could illustrate these situations. Two systems would be maintained: a sales personnel system, which might have fields for name, employee number, quota, percentage commission, and base salary; and an order system with fields for date, customer, item, quantity, price, amount of sale, and employee number of the person who made the sale. Thus there would be many records in the order system corresponding to each record in the personnel system. With the sales personnel system as base, there are two choices for linking the order system: by **totalling**, or by **replicating**. The following example shows the linkage using totalling to associate with each employee a field giving total sales in April.

```
10:LS
* LINK SYSTEM
* CREATOR AND SYSTEM NUMBER:  20
* TOTALLING REQUIRED? (Y/N):  Y
* KEY FIELD ON CURRENT SYSTEM:  SLSNUM
* KEY FIELD ON LINKED SYSTEM:  SLSNUM
* LINK SYSTEM RECORD CONSTRAINT: DATE=0481
     2400 FOUND.
* FIELDS TO LINK:  AMOUNT
* NEW FIELD NAMES: APRILSALES
* OK TO LINK (Y/N): Y
* SYSTEMS LINKED.
```

Note that the constraint ensures that only orders for April 1981 are considered, and the totalling option is specified. Thus the *AMOUNT* field for these orders will be subtotalled for each corresponding record in the sales personnel system in a new field named *APRILSALES*.

Conversely, by entering the order system and linking the sales personnel system, a field from the personnel system, such as commission rate, can be associated with each sales order by replicating fields from the linked data base. Thus you could easily calculate total commissions in the sales order data base.

A New Type of Field

Another new feature of MABRA is **pseudo fields**. Pseudo fields are 'almost' fields in the usual MABRA sense. They are calculated from real MABRA fields, and can be printed on reports, linked, or used in constraints just like real fields. However, they are not stored in the data base, but are calculated when referred to. In every other way they are indistinguishable from real fields.

In the order system described above, the field *AMOUNT* is an example of a pseudo field which can be calculated as *QUANTITY* × *PRICE*.

Pseudo fields are added to a system by using the new *AP* (**Add Pseudo**) command. The following session would add the pseudo field *AMOUNT* to our sales order data base:

```
20:AP
* ADD PSEUDO FIELD
* ENTER FIELD NAME:  AMOUNT
* TYPE? (CHAR/INTE/REAL/BINA/DATE): INTE
* NUMBER OF ELEMENTS IN FIELD:  1
* USE DEFAULT OUTPUT FORMAT? (Y/N):  Y
* RETAIN THIS FIELD? (Y/N):  Y
* ENTER EXPRESSION DEFINING FIELD
:QUANTITY×PRICE
* OK TO ADD FIELD? (Y/N):  Y
* FIELD ADDED.
```

Users familiar with MABRA will notice that adding a pseudo field is similar to adding a real field using the *AF* command. Notice that, as creator, you have a choice of retaining the pseudo field, or having it deleted automatically as soon as you terminate the MABRA session by leaving system mode.

Pseudo fields can be used to derive numeric field values as the result of an arithmetic relationship between other numeric fields. They can also be used for character fields. For example, when names are stored in separate fields for first name, middle initial, and last name, and you want to print the full name, you can define a pseudo field to combine the three name fields into one, squeezing blanks, together with an appropriate salutation based on sex. Furthermore, since any APL expression is valid, extremely sophisticated expressions can be built which might even retrieve data from some other private or public data base.

New Reporting Features

Another major extension to MABRA is the addition of a report generating command *GR* (**Generate Report**). *GR* generates a report similar in format to List Record (*LR*), but subtotals are generated. Subtotals are calculated after each change or 'break' in the sort keys you specify before the slash. Sort keys listed after the slash generate no subtotals. Suppose you want to generate a report which lists each employee's name, sex, EEO (ethnic description) and salary, and you want to subtotal salary for each sex/EEO combination. Your session would appear as follows:

```
10:GR
* GENERATE REPORT
* CONSTRAINTS:  ⍺DOHIRE=10
     21 FOUND.
* PRINT FIELDS:  SEX,EEO,NAME,SALARY
* SORT FIELDS:  SEX,EEO/
* TOTAL FIELDS:  SALARY
* PRINT RECORD DETAILS? (Y/N): Y
* ANY SPECIAL OPTIONS? (Y/N): N
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
DATE:  14/04/81                                        PAGE 1
                        40: PERSONNEL SYSTEM

SEX   EEO                  NAME                      SALARY

F     BLACK                SLAUGHTER,PHYLLIS         86,711
F     BLACK                WATKINS,DOLORES           17,341
                                                     ---------
F     BLACK                                         104,052

F     HISPANIC             MARTINEZ,IRENE            21,018
                                                     ---------
F                                                   125,070

M     BLACK                SMITH,ADAM                76,272
M     BLACK                MARTIN,JOHN               46,992
                                                     ---------
M     BLACK                                         123,264

M     HISPANIC             RODRIGEZ,MIGUEL           79,856
M     HISPANIC             CANDIANI,VICTOR           76,082
M     HISPANIC             LOPEZ,VICENTE             86,767
M     HISPANIC             MENDOZA,ALEJANDRO         31,394
                                                     ---------
M     HISPANIC                                      274,099

M     INDIAN(AMERICAN)     PARADISE,SAL              13,000
M     INDIAN(AMERICAN)     BLACKFOOT,LOUIS           22,000
M     INDIAN(AMERICAN)     MEANS,RUSSELL             99,829
M     INDIAN(AMERICAN)     BROOK,JOHN                62,281
M     INDIAN(AMERICAN)     GREYBEAR,LEWIS            36,835
M     INDIAN(AMERICAN)     WOOLEY,MATTHEW            83,792
M     INDIAN(AMERICAN)     COUSINS,MARTIN            56,597
M     INDIAN(AMERICAN)     GILMORE,STANLEY           10,000
M     INDIAN(AMERICAN)     BROWN,WILLIAM J.          25,483
                                                     ---------
M     INDIAN(AMERICAN)                              409,817

M     WHITE                CUPPS,FRANK               90,997
M     WHITE                BOOK,LAURENCE             49,351
M     WHITE                STOTT,CHARLES             83,624
                                                     ---------
M     WHITE                                         223,972
M                                                   1,031,152
                                                    =========
                                                    1,156,222
```

Note that the constraint α*DOHIRE*=10 finds all employees who have been with the company for ten years. The *GR* listing is similar to List Records (*LR*), except that salary subtotals are printed for each sex category, male and female, and sub-subtotals are printed for each sex/EEO combination. In addition, a salary grand total is printed. Note that you have the option of suppressing record details, in which case only subtotals for each sex/EEO combination are printed.

Other New Features

In addition to the features described above, several other enhancements have been made to MABRA as well. *DL* and *PL* commands provide for the definition and printing of custom-specified listings interactively. Reports can now be generated at your terminal with paging, page numbering, and user-specified titles. The *AN* crosstabulation analysis command now allows you to request percentages based on row, column or grand totals. *LF*, List Fields, has an option to print more detail about field definitions and field size. State-setting commands have been added so that dates can be printed in U.S. format, and for a new type of search constraint for those users who prefer to specify APL constraints.

All of these features are available in a new release of MABRA in 586 *MABRA* and 585 *MABRAUTIL*, and on-line documentation can be obtained by typing *NEWFEATURES* in either workspace. This preliminary release of MABRA is intended to give you an opportunity to give us feedback on these new features, and we encourage you to send your comments to us, either through your local branch or through the mailbox code *MABQ*. The software will be installed in 86 *MABRA* later this year.

APPLICATIONS LIBRARY UPDATE

New:

1 *APL82* - This workspace was installed to facilitate the submission of abstracts for APL82 (see page 8).

Changed:

1 *FILEAID* - The function *TPRINT* from 1 *HSPRINT* has been added.

1 *FORMAT* - A new release (see page T1).

1 *REFERENCE* - The tables have been updated to reflect recent additions to SHARP APL.

Gone:

The following workspaces have been superseded. They are now in library 499.

14 *APLSTARTER* (use 5 *FONT*)

23 *LPAPL* (use 43 *LINPROG*)

23 *STP4* (use 48 *SNAP*)

23 *STP5*, 23 *STP6* (use 543 *TRANSPORT*)

FAR EAST DIVISION

I.P. Sharp Associates will begin their expansion into the Far East with the opening of the Singapore office on May 15. The Hong Kong office will open on July 15th. The expansion offers our customers faster, more effective communications. Formerly effected by telephone or mail, communications will be virtually instantaneous using the I.P. Sharp network.

Walter Keirstead heads the operation as the company's area representative. Walter, who has his M.Sc. and B.Sc. from Laval University, has been with I.P. Sharp Associates as Branch Manager of the Montreal office since 1973. He has 16 years experience in planning, investment analysis, cost control, and forecasting.

Walter Keirstead and Ian Sharp

SINGAPORE

Hugh Hyndman is the Branch Manager of the new office in Singapore. He has his B.Sc. in Computer Science from the University of Toronto. Since joining I.P. Sharp in 1976, he has worked in Toronto as an account representative, in London, England on corporate financial systems development, and in Birmingham, England, as Branch Manager. In late 1979, Hugh returned to Toronto to concentrate on the development of MABRA and other new applications software.


Hugh Hyndman

Hugh Hyndman will be assisted in Singapore by **Mark Seltzer**. Mark has a Bachelor of Integrated Studies degree from the University of Waterloo in Canada. Since joining I.P. Sharp Associates in 1979, he has worked for the Data Base group on several data base projects. His primary involvement has been in the development and enhancement of the SUPERPLOT package. In Singapore, his main responsibility will be customer support and application systems development.


Mark Seltzer

technical supplement 32

The existing workspaces on the SHARP APL system, some of which have now been around for several years, do not normally feature in the newsletter because, by definition, they are not 'news'. However, they are the staple diet of the SHARP APL user. In this series of articles various contributors will review their favourite 'old' system features and workspaces. It should be possible for users to judge from the articles their potential usefulness in their own applications. If you would like to contribute to this series please contact Lib Gibson in our Toronto office.

PRODUCING REPORTS TO SPECIFICATION QUICKLY!

Workspace 1 *FORMAT* Revisited

James Sinclair, London

Most commercial APL applications produce reports: the final outputs of a system neatly formatted on the page with such adornments as titles, column headings and row labels. Designing the report format can be something of an art, but once decided the effort required to make the computer reproduce the design should, naturally, be minimised. This is particularly true if writing a program for someone else (the 'user'). It is usually good practice to code the report-writing sections of a program as early as possible. Firstly, it helps you (the APL practitioner) to verify that you have understood the user requirement. Secondly, it gives the user confidence. Thirdly, once users see the reports they usually request changes. If you can modify the reports quickly (possibly on a while-you-wait basis) you will not only greatly impress the user, but you will probably arrive at a much more accurate reflection of the real requirement.

One approach is to use a program with report-writing capabilities such as MAGIC. However, there will be times when you may wish for the precision of a lower level tool. In these cases ⎕*FMT* and its frequently forgotten brother workspace 1 *FORMAT* are appropriate. (Incidentally, the workspace has recently been refurbished by Ken Pawulski).

⎕*FMT* The Formatting Function

If you are not familiar with ⎕*FMT*, it is well worth finding a couple of hours in which to learn its operation. It is the most exact formatting aid on the system, and is extremely powerful.

As a very basic example, take a variable *PERCENT* which contains percentages. We wish to display the array, each element to occupy **four** positions on the paper, shown to the nearest **integer** (no decimal point), with the suffix ∘/∘ attached to each number.

```
      PERCENT
38.6 38.3 21   26   56
40    0   32.3 0    26.4

      'I4,<∘/∘>' ⎕FMT PERCENT

 39∘/∘  38∘/∘  21∘/∘  26∘/∘  56∘/∘
 40∘/∘   0∘/∘  32∘/∘   0∘/∘  26∘/∘
```

The right argument is the variable to be formatted, whilst on the left we have the format specification. We can 'qualify' the format phrase *I4* by requesting (for example) that zero values be left blank (*B*-qualifier). If there were large numbers present, we could specify that commas should delimit the thousands (*C*-qualifier). So we might have:

```
      'BCI4,<∘/∘>' ⎕FMT PERCENT
```

Needless to say, this is a very basic use of ⎕*FMT*. For more information, see the manual *SHARP APL Report Formatting*. The functions *FMTEXAMP* and *DEMO* in workspace 1 *FORMAT* are useful self education tools; they contain examples of ⎕*FMT* in action.

Workspace 1 *FORMAT*

Whilst ⎕*FMT* will format the body of a report, several functions in workspace 1 *FORMAT* help produce titles, report labels and column headings. A description of each function exists in 1 *FORMAT* as a variable of the same name with *HOW* tacked on the end. For example, a description of the *COLNAMES* function may be obtained by typing *COLNAMESHOW*.

To copy the functions alone into your workspace, type:

```
)COPY 1 FORMAT FMTAID
```

As an example, assume we were to produce the following report for a mythical chocolate manufacturer:

LONDON BRANCH MARCH SALES
CASH VALUES IN STERLING (UKL)

PRODUCT	PRICE (UKL)	SALES (BARS)	TARGET (BARS)	VARIANCE (UKL)	VARIANCE (∘/∘)	REVENUE (UKL)
ASTRABARS	0.15	28,583	30,000	(213)	-5	4,287
PLAIN BARS (100 GMS)	0.25	36,522	30,000	1,631	22	9,131
PLAIN BARS (250 GMS)	0.50	15,089	15,000	45	1	7,545
MILK BARS (100 GMS)	0.25	58,653	50,000	2,163	17	14,663
MILK BARS (250 GMS)	0.50	17,821	18,000	(90)	-1	8,911
		156,668	143,000	3,536	7	44,536

The data is available in variables *PRODNAMES* (product names), *PRICE*, *ACTUAL* (sales) and *BUDGET* (the target sales).

Use ⎕*FMT* to produce the body of the report (excluding headings and totals):

```
FS←'20A1,BF7.2,2CI9,M<(>N<)>CI10,M<->I9,CI9'
FS ⎕FMT (PRODNAMES;PRICE;ACTUAL;BUDGET;PRICE×
          ACTUAL-BUDGET;(100×ACTUAL
          ÷BUDGET)-100;ACTUAL×PRICE)
```

Although it appears complicated, the above is really quite simple once you learn to read ⎕*FMT* statements. The right argument contains the data to be formatted. In this case some of the items are calculated. Each row is then formatted according to the format specification *FS*. The exact meaning would be understood after some reading and experimentation with ⎕*FMT*. Essentially, the *F7.2* means a number, with a width of 7 positions, two digits after the decimal point and 4 before. *M<(>* requests parentheses around negative numbers, while *M<->* requests a low minus before negative numbers.

We might well have produced *PRODNAMES* (the matrix of product names) using the 1 *FORMAT* function *ROWNAMES*. Most APL people have a similar function tucked away somewhere; *ROWNAMES* produces a character matrix from its right argument. For example:

```
      PRODNAMES←20 ROWNAMES 'ΩASTRABARSΩPLAIN
         BARS (100 GMS)ΩPLAIN BARS (250 GMS)Ω
         MILK BARS  (100 GMS)ΩMILK BARS  (250 GMS)'
      PRODNAMES
ASTRABARS
PLAIN BARS (100 GMS)
PLAIN BARS (250 GMS)
MILK BARS  (100 GMS)
MILK BARS  (250 GMS)
```

Here *PRODNAMES* will have a width of 20 (the left argument) with each row filled with one of the names on the right. The first character of the right argument is the delimiter.

The functions *COLNAMES* and *CENTER* (*Note: this may not be your local spelling*) can be used respectively for column headings and titles. Both functions take the format specification being used as the left argument. In our case we refer back to *FS* shown above. *CENTER* analyses the format specification and centers the right argument over the space that data applied to *FS* through ⎕*FMT* would occupy. *COLNAMES* analyses the format specification and places column headings from the right argument over the columns that would be occupied by data applied to *FS* through ⎕*FMT*.

```
FS CENTER 'LONDON BRANCH MARCH SALES'
FS COLNAMES 'ΩPRODUCTΩPRICEΩSALESΩ
    TARGETΩVARIANCEΩVARIANCEΩREVENUE'
```

Thus: to produce the report above I included the following statements in a function.

```
FS←'20A1,BF7.2,2CI9,M<(>N<)>CI10,M<->I9,CI9'
FS CENTER 'LONDON BRANCH MARCH SALES'
FS CENTER 'CASH VALUES IN STERLING (UKL)'
' '
FS COLNAMES 'ΩPRODUCTΩPRICEΩSALESΩTARGETΩVARIANCEΩ
             VARIANCEΩREVENUE'
FS COLNAMES 'Ω Ω(UKL)Ω(BARS)Ω(BARS)Ω(UKL)Ω(∘/∘)Ω(UKL)'
' '
FS ⎕FMT(PRODNAMES;PRICE;ACTUAL;BUDGET;PRICE×
       ACTUAL-BUDGET;(100×ACTUAL÷
             BUDGET)-100;ACTUAL×PRICE)
FS COLNAMES 'Ω Ω Ω-------Ω-------Ω------Ω--Ω------'
FS ⎕FMT(1 20 ρ' ';0;+/ACTUAL;+/BUDGET;+/PRICE×
             ACTUAL-BUDGET;(+/(100×ACTUAL÷
             BUDGET)-100)÷ρACTUAL;+/ACTUAL×PRICE)
FS COLNAMES 'Ω Ω Ω=======Ω=======Ω======Ω==Ω======='
```

The report took only a few minutes to code. Note that should we decide on minor changes, such as changing the spacing between the columns (as in fact I did), this can be done with minimal impact to the function. Just repecify *FS* and everything else falls into place.

It is true that the functions *CENTER* and *COLNAMES* analyse *FS* every call. There are economies to be achieved by eventually replacing

COLNAMES and *CENTER* once the report has been finalised (if it ever is).

If you find yourself writing reports frequently, then we strongly recommend a look at ⎕*FMT* and workspace 1 *FORMAT*. If you have any problems, ring up your local I.P. Sharp office. To misquote one of the earliest books on APL: 'The descriptive and analytic power of an adequate programming language amply repays the considerable effort required for its mastery.' That applies to ⎕*FMT* and 1 *FORMAT* as much as any other part of the language. The only part of of the statement to be questioned is the word 'considerable'. Two hours should prove a sufficient introduction.

SYSTEM VARIABLES, Part I

Robert Metzger, Rochester

System variables are unsung heroes in the world of APL. Everyone takes them for granted. Yet if you must work in another computing environment, you will quickly come to appreciate their value.

System variables are a special kind of shared variable. They are shared between a task and the APL system. The sharing happens each time a workspace is activated. It also happens each time a defined function is executed, if it has system variables localized in its header.

System variables can be split into several groups.

Output Control	- ⎕*PW*, ⎕*PP*, ⎕*HT*
Computational Control	- ⎕*CT*, ⎕*IO*, ⎕*RL*
Event Control	- ⎕*LX*, ⎕*ER*, ⎕*TRAP*
Miscellaneous	- ⎕*SP*

Some APL systems have implemented ⎕*AI*, ⎕*AV*, ⎕*LC*, ⎕*TS*, ⎕*WA*, ⎕*UL* as Informational system variables. Although they can be assigned by a defined function, the APL system does not use the value assigned. They are always reset by the APL system, which is why they are niladic system **functions** in SHARP APL.

The purpose of this article is to describe some common uses of the Output and Computational Control system variables. We will begin by looking at the Output Control system variables. APL is one of the few computer languages around which will display the result of an expression if you simply execute the expression. Some languages do provide a free form output capability. But even these usually require you to use an explicit command to produce the output. That APL can provide default output is due, in part, to these system variables.

⎕*PP* stands for Print Precision. It determines how many digits will be displayed to the right of the decimal point in default output. It must be an integer between 0 and 18. Its default value is 10. Its effect on default output is shown below.

```
      ⎕PP←10 ◇ 2÷3 ◇ ⎕PP←16 ◇ 2÷3
0.6666666667
0.6666666666666667
```

Remember, ⎕*PP* controls the number of digits **displayed**, not the number of digits **carried**. While it affects default output and monadic ⍕, it does not affect ⎕*FMT* and dyadic ⍕. They explicitly control digit display.

When would you want to change the value of ⎕*PP*? Sometimes you want to display a variable at the terminal, but you don't need to see all the digits to the right of the decimal point. You can speed up your display by setting ⎕*PP* lower than the default value, perhaps ⎕*PP*←5. Another situation in which changing ⎕*PP* is useful, is when you are dealing with very large numbers. When an element in an array is big enough, it will be displayed in scientific notation, even if it is an integer. You can force the system to use the standard format by increasing ⎕*PP*, usually to its maximum value. The examples below show this behaviour.

```
      ⎕PP←10 ◇ 1234567890123 ◇ ⎕PP←16 ◇ 1234567890123
1.23456789E12
1234567890123
```

⎕*PW* stands for Print Width. It determines the maximum number of characters per line transmitted to a terminal in default output. It must be an integer between 30 and 250. Its default value is 132. Several common values are listed below.

80	Video Display (CRT)
132	Printer showing 10 characters per inch
156	Printer showing 12 characters per inch

⎕*PW* has several happy effects. When a display will take more than ⎕*PW* positions, it indents continuation lines six spaces from the left margin. This makes it possible for you to distinguish continuations from distinct rows of a matrix.

When numbers are displayed as default output, the system 'folds' the output between numbers, ie. at blanks. Try the following line to see the difference between default numeric and character output.

```
      ⎕PW←30 ◇ ⍳100 ◇ ⍕⍳100
```

When functions are displayed using the ∇ editor, names and numeric constants are **not** split. This accounts for the difference you sometimes see when you do ∇*FUN*[□]∇ versus 1 □*FD* '*FUN*'. The latter expression produces character output, and is not given special treatment.

What can you use □*PW* for besides making your output fit your terminal? One example would be when you have a matrix with a lot of rows you would like to display. You can reduce the amount of paper required by printing your matrix in multiple columns. A program can maximize the amount of data displayed within □*PW*.

```
    ∇ MAT←MATACROSS MAT;ROWS;COLS
[1]   MAT←' ',⍕MAT
[2]   COLS←⌊□PW÷¯1↑⍴MAT
[3]   ROWS←⌈(1↑⍴MAT)÷COLS
[4]   MAT←(ROWS,COLS×¯1↑⍴MAT)⍴((ROWS×COLS),¯1↑⍴MAT)↑MAT
    ∇
```

A function which displays the data down the page would use □*PW* in the same way. It would have the following differences.

```
    ∇ MAT←MATDOWN MAT;ROWS;COLS
[4]   MAT←⍉2 1 3⍉(((1↑⍴MAT)÷ROWS),ROWS,¯1↑⍴MAT)⍴
          ((ROWS×COLS),¯1↑⍴MAT)↑MAT
[5]   MAT←((1↑⍴MAT),×/1↓⍴MAT)⍴MAT
    ∇
```

□*HT* stands for Horizontal Tabs. This system variable makes it possible for the APL system to take advantage of the tab stop features of many terminals. Unlike □*PP* and □*PW*, it affects both input and output. The default value is ⍳0, which means no tabs. It can be set to any integer scalar or vector. Each integer in □*HT* must be less than or equal to □*PW* and greater than or equal to 0.

When you set □*HT*, the APL system assumes that you have also set corresponding tab stops on your terminal. You can do this manually. On some terminals, you can also do it automatically, by sending the necessary characters through □*ARBOUT*.

The purpose of □*HT* is to reduce the number of characters transmitted to or from a terminal. This means easier entry, faster printing, and less cost. When you press the TAB key, the system converts that character into the number of blanks between the position you were at, and the next tab stop. So 1 tab character is transmitted, but it is interpreted as 1 or more blanks. When you print blank spaces, the system converts groups of blanks into a single tab character where possible. So fewer characters are sent than appear on your display.

How you set your tabs depends on whether you are doing input or output. For output, you probably want tabs set at even intervals. This maximizes the probability that a group of blanks can be converted to a tab character. For input, you will probably want to set a tab at the beginning of each field. This will minimize keystrokes.

In SHARP APL, you tell APL that tabs are set at even intervals by assigning the interval, as a scalar, to □*HT*. If you want to tell APL that tabs are set at specific positions, assign a vector of the positions to □*HT*.

Don't forget that not only do we have to tell the APL system that tabs are set, but we have to set them on our terminal. If our terminal interprets the ESCAPE character, followed by some designated character (often '1') as a command to set a tab stop, the following code will work for fixed intervals:

```
□ARBOUT ((2+INTERVAL)×⌊□PW÷INTER
      VAL)⍴(INTERVAL⍴SPACE),ESCAPE,¯1+TRANSLATE⍳TABSET
□HT←INTERVAL
```

For ASCII terminals, *SPACE*←32 ◇ *ESCAPE*←27 and *TABSET* is a terminal dependent value. *TRANSLATE* is either *ARBBIT* or *ARBTYPE* from workspace 1 *ARBOUT*, depending on whether your terminal is 'Bit Paired' or 'Typewriter Paired'.

For setting designated positions, we have a different approach.

```
TABS←((□PW⌊⌈/POSITIONS)+2×⍴POSITIONS)⍴SPACE
TABS[1+,⍉0 1∘.+POSITIONS+2×¯1+⍳⍴POSI
      TIONS]←(2×⍴POSITIONS)⍴ESCAPE,TABSET
□ARBOUT TABS
□HT←POSITIONS
```

In both of these cases, the system function □*ARBOUT* is used to transmit control codes. □*ARBOUT* sends the binary equivalent of its decimal arguments to the terminal. None of the APL Input/Output formatting software is used. Thus, the responsibility for sending codes meaningful to the terminal is entirely left to the APL program.

System variables play an important part in the APL default output feature. They can be used to modify that feature to meet special application needs. In the next part of this series, we will explore how system variables can be used to modify the behavior of primitive functions.

CANBERRA

Dr. Ahnont Wongseelashote has been appointed Branch Manager of the new Canberra office. He comes to I.P. Sharp Associates from Bangkok. Ahnont was sponsored by the Bank of Thailand, and has a B.Sc. (Mathematics) and an M.Sc. in Economics (Operational Research) from the University of London, School of Economics and Political Science, and a Ph.D. from the University of Southampton, England. For four years prior to joining I.P. Sharp, Ahnont was with the Bank of Thailand in the Economics Research department, where he gained valuable experience as a systems analyst and APL programmer. He has also designed information systems in APL, and provided support to economists in using time series techniques and other statistical methods for economic forecasting. He is keen to continue his work in the areas of operations research, statistics and financial modelling.

Dr. Ahnont Wongseelashote

LOS ANGELES

I.P. Sharp now has an office in Los Angeles, situated between Beverly Hills and West Los Angeles. **Frank Mullin** is the consultant on staff there.

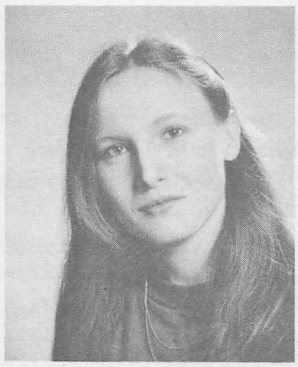
The new office is the fifth I.P. Sharp office in California, Los Angeles and Newport Beach in the south, and San Francisco, Palo Alto and San Jose in the north.

NOTE

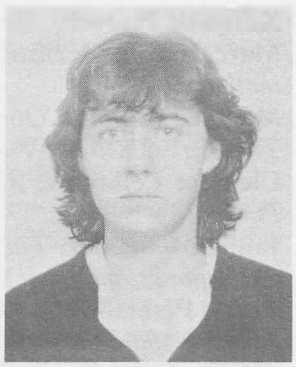
The Semantics of Air Passenger Transportation was published by the Norfolk Port and Industrial Authority. Copies are available from the distributor, ISGS ($19.95 postpaid) at 834 Mission St. (4th floor), San Francisco, Ca. 94103.

PARIS

Nelly Detre joined the Paris office from the APL team at Citroen. With a financial background and several years experience in APL, she has concentrated on financial applications and APL education. She contributed the review of the Supelec conference below.



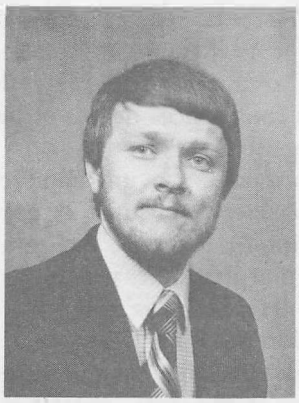
Nelly Detre



Nathalie Merlin

Nathalie Merlin has a statistics background and is continuing her studies at the INSEE (French National Statistics Institute). She has spent some time implementing a French Correspondence Analysis method in SHARP APL, which has proved to be successful in the analysis of large quantities of data.

KITCHENER-WATERLOO



Brian Olson

A new office has been established by **Brian Olson**. Brian received a Bachelor of Commerce degree from the University of Toronto in 1976. He spent 3 years with a Canadian automotive distributor where he first learned APL. In April 1979 he joined I.P. Sharp Associates where he has been involved with the design and marketing of the Lease Evaluation System.

CONFERENCES

WESTERN APL

The Four Seasons Hotel, Edmonton
June 4, 1981

Come to Edmonton on the first Thursday in June for a meeting of APL-minded people. The registration fee is $50. A light lunch will be served, and a reception is planned at 5pm. You will hear:

Kenneth Iverson, *Managing APL*
Marc Baron, *A Distribution Management Information System*
Maurice Elliott, *'Open' vs 'Closed' systems*, or *The Art of Killing Two Birds with One Stone*
Clement Leibovitz, *APL Interface to Fortran and Portability*
Jaime Menendez, *The Use of a System to Handle Operations Planning Models*
Terry Peterson, *The Use of MABRA for Monitoring Capital Project Expenditures*
Phillip J. Rody, *A Stars Application in Multicorporation Financial Planning*
Lloyd Sereda, *Managing Resistance and Expectations in the Implementation of a New System*
Walter Yarish, *Computerization of Weed Control Recommendations*
Paul Berry, *Structure and Style in APL Programs*

FORUM SUPELEC

Nelly Detre, Paris

SUPELEC, (Ecole Superieure d'Electricite), une des grandes ecoles Francaises, specialisée dans le formation d'ingenieurs electroniciens et informaticiens, organisait les 18 et 19 Mars 1981 un forum sur l'informatique, comprenant des stands d'exposition ainsi que des conferences debats et les profils de carriere des ingenieurs.

I.P. Sharp y etait representē par un stand et avait pour voisins des societes francaises importantes telles que Elf-Aquitaine, Renault, Aerospatiale ..., ainsi que des enterprises de production informatique telles que Hewlett-Packard, Burroughs, Control Data, Matra ...

Les visiteurs, furent tout specialement interessés par nos systemes de type graphique, (SUPERPLOT, Graphics), ainsi que par notre rēsau international de communications, et furent tres impressionēs par les reponses instantanées (en provenance de Toronto!!!) comparēes aux temps de reponse souvent longs de leur systēme situe dans l'ecole.

Beaucoup ont decouvert APL, car ils realisent leurs applications principalement scientifiques en Fortran et Basic, et ont apprecié le caractēre synthetique et matriciel de ce "nouveau" langage.

CALL FOR ABSTRACTS

APL 82 Heidelberg, Germany
July 26-30, 1982

Sponsored by the
APL CLUB GERMANY
and
Deutches Krebsforschungszentrum, Heidelberg
in cooperation with
ACM/SIGAPL

Abstracts should contain about 100 words and a representative description of the intended content of the full paper. Please submit them **before 1 September 1981**, in English, to:
Prof. Dr. Wolfgang H. Janko
Program Chairman APL 82
Universitaet Karlsruhe (T.H.)
Institut fuer Angewandte Betriebswirtschaftslehre
Am Zirkel 2 (Rechenzentrum, 1.OG),
Postfach 63 80 D-7500 Karlsruhe 1/Germany

PUBLICATIONS

APL/STAT

By J.B. Ramsey & G.L. Musgrave
(Lifetime Learning Publications 1981)

Reviewed by: Mike Powell, Victoria

Did you ever wonder what was inside the dark recesses of those 'black box' statistical packages? Or about JCL and FORTRAN subroutine libraries? Did such tools hinder your understanding of statistics? If so, you will be interested in reading this new book by Ramsay and Musgrave.

The authors have provided a bridge between formal statistical instruction and the application of statistics to real problems. The relevant statistical theory is not included in the book; it is assumed that readers are already familiar with the theory, or are learning it concurrently. All the APL required is thoroughly explained in the text.

The early chapters deal with basic statistics, linear regression, and ANOVA, while later chapters explain the more popular techniques used in econometric modelling (including two stage least squares, instrumental variables, and LIML). Each chapter has exercises (and answers) and these give the reader ample opportunity to exercise his skills.

This book will certainly be read often, both by those currently concerned with statistical calculations, and as a text to accompany a more formal course in statistics. Statistics can be painless!

THE PSYCHOLOGY OF WRITING PROGRAMS

Peter Airs, London

For many people the first experience of programming is a terrifying ordeal. Novices are often subjected to a barrage of strange terminology, buzzwords and esoteric numbers like 370, 3330 and 200 Megs. Before long they are submerged under a mountain of computer printouts comprising a dazzling array of totally obscure error messages and core dumps. After 2 weeks they emerge white haired, bemused and still none the wiser. This is often the fate of the unsuspecting university undergraduate who is subjected to the obligatory lightning course in FORTRAN or ALGOL, and it can result in complete alienation from programming.

It comes as quite a shock when these same people discover that they do possess considerable programming skills, as frequently happens when they encounter APL. The opposite experience is also common. Graduates who emerge with flying colours from university courses such as Computer Science, sometimes discover to their horror that they are useless in a commercial programming environment.

The Best Language?

Most computer programmers think that they select the languages in which they program. However, it is equally true to say the opposite .... computer languages 'choose' the people that use them. Languages such as FORTRAN COBOL ALGOL and APL will appeal to a programmer with a particular profile of mental skills and psychological makeup. The characteristics that make up a successful COBOL programmer may prove fatal to programming in some other language such as ALGOL or APL. The converse is also true, since APL favours different mental abilities to COBOL. Being a failure at one computer language does not mean that you are doomed to fail in another language.

To be successful at COBOL, for instance, you need great powers of concentration and perseverance. COBOL dictates an approach where you write monolithic chunks of code which must work. The mammoth task of coding several thousand lines of code is a test of endurance requiring a strong writing arm and lots of HB pencils. Making a hash of things in COBOL is terribly time consuming and potentially catastrophic. You have to plod your way through COBOL, slowly and meticulously.

In contrast, APL allows a more volatile, experimental and flamboyant approach. You chip away at the the problem by writing many discrete functions. APL very graciously allows you to correct mistakes with the minimum of distress. You can write programs on a trial and error basis, where you juggle APL primitives until the desired result is achieved, and then incorporate them into the body of the program. This feature is very important to the inexperienced programmer.

Since all operations in APL can work on multidimensional data objects, it is useful to be able to visualise data arrays in two, three and more dimensions. This visual-spatial ability helps one to understand APL operations such as those which rotate, transpose and reshape data.

The 'best' language is the one that suits you. In general a good APL programmer will not make a good COBOL programmer, and vice versa. Not all languages are conceptually as far apart as COBOL and APL. Many are very similar to another language, for example PASCAL to ALGOL, and BASIC to FORTRAN.

Systems Design in APL

The characteristics of a computer language reach out far beyond writing computer code. Language is the means that you employ to *think* and *perceive* the environment as well as the means to *express* yourself. All problems solved with the aid of a computer are seen and evaluated in terms of the computer language(s) in view. It is always the **perceived problem** that is solved. The perceived problem may be 100 times more difficult in one language than in another more suitable language, and correspondingly more expensive.

An illustration of the difference in effort required in different languages is the familiar problem of generating an *explosion* of a hierarchy, such as, 'tell me all the individual parts required to build a car?'. In many languages this is achieved by an enormous amount of code and the creation of work files which link records together with all manner of pointers, chains and flags. In a language which allows recursive programming (APL) the same explosion can be programmed very simply in a few lines of code.

The choice of computer language will have dra-

matic effects on systems design as well as computer code. In order to specify a good APL system the systems designer needs to have a good appreciation of APL. Specifying an APL system while thinking in COBOL mode does not produce good results. Fortunately, APL requires much less detailed specification work than most other languages. Often hefty tomes of computer specifications, produced at enormous cost by COBOL orientated analysts, can be safely ignored or quietly forgotten if the system is to be implemented in APL.

Traditionally, systems design was thought of as an *analytic* process, whereas the experience with APL systems design is that the process is both *analytic* and *synthetic*. Analytic design processes require you to break down a problem into series of familiar steps. But what if there is no handy precedent?. Here rigor-mortis rapidly sets in and the problem is usually ignored. APL offers a more dynamic and experimental approach to systems design where the design can be created or synthesised, usually by trial and error. After several iterations you gain an appreciation for the data on which the final design is based. To paraphrase a well known expression ... APL allows you to boldly go where no programmer has been before.

Creativity in APL

One of the reasons why it is easy to be creative in APL is the close relationship between thinking in APL and creative thinking: the generation of new concepts by juggling with existing ideas. The mind can deal with a limited number of concepts at any particular time. Because data operations in APL are usually done on the variable as a whole, rather than on each of its elements in turn, it is easier to conceptualise what is happening to data. This helps the creative process. Other languages that force you to consider operations on an element by element basis impose an extra level of complexity. The result is the computing equivalent of 'You can't see the wood for the trees'.

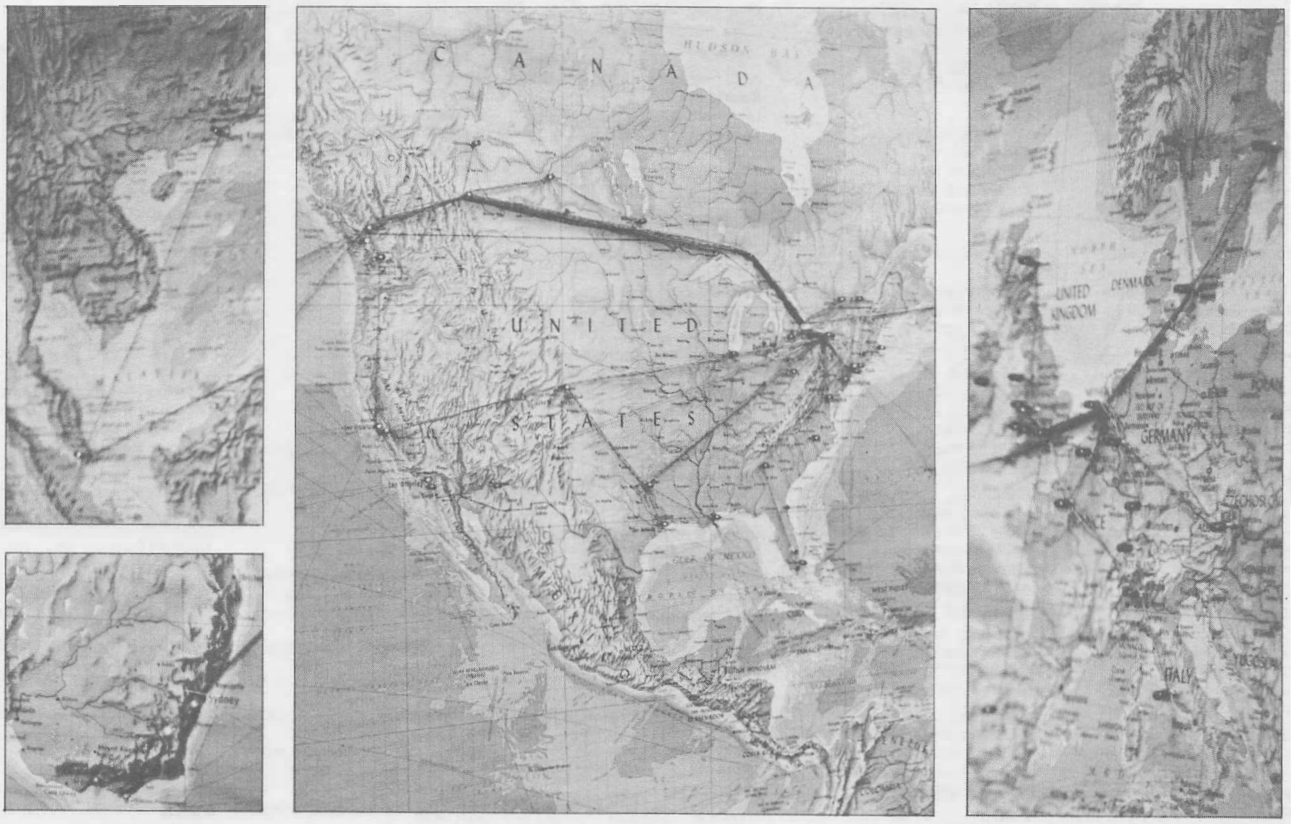
You can conceive an APL variable as a whole because both the structure (or shape) and data values are stored together. The shape allows you to perceive the data in a very concrete way. It is useful to consider two or three dimensional objects as you would the slotting together of building bricks in a LEGO set. This visual, almost mechanical technique to picture APL arrays is a skill which may be the heart of the creative thinking process. Koestler, in his book *The Ghost in the Machine* says, 'Language can become a screen between the thinker and reality; and creativity often starts where language ends, that is, by regressing to pre-verbal levels of mental activity'. As evidence he quotes one of the most original thinkers of our time, Einstein, "The words of language as they are written or spoken do not seem to play any role in my mechanism of thought, which relies on more or less clear images of a visual and some of a muscular type."

Most recent computer languages, with the notable exception of APL, have been built around the English language. They are full of English reserved words such as FOR WHILE DO IF THEN ELSE. The argument for this is that English is a *natural language* and thus aids the thinking process. What Koestler is saying is the opposite, that the creative thinking process uses a more conceptual and visual approach. APL, a mathematically based language, provides the programmer with a language which is much closer to creative thinking and as such it could be said to be a more natural language than all the English based ones.

As a piece of pure speculation it might be interesting to consider the theory on the difference between the two hemispheres of the brain. Research has indicated that one hemisphere (usually the left) is responsible for analytic processes, while the other (right) is responsible for synthetic processes. Programming tends to be a very linear, analytic process. One could say that APL, with its ability to encapsulate very complex ideas in a small number of symbols, gives the synthetic (right) hemisphere much more scope. If this is the case then APL obviously makes more efficient use of our mental resources than other computer languages.

In conclusion, the success of a computer language depends on how programmers use the language to describe the problem they are trying to solve. The efficiency of the problem description will depend on how useful the concepts or idioms of the computer language are. While many programming languages have evolved a set of useful idioms to *analyse* a particular problem, APL also provides a set of tools to *synthesise* new solutions for hitherto unsolved problems.

I.P. Sharp Communications Network



MAILING REQUEST

- ☐ Please amend my mailing address as indicated.
- ☐ Please add the following name(s) to your Newsletter mailing list.
- ☐ Please send me a Publications Order Form.
- ☐ Please add my name to the Energy Newsletter mailing list.
- ☐ Please add my name to the Aviation Newsletter mailing list.
- ☐ Please send me information about your courses in

(city) _____

Name: _____

Title: _____

Co.: _____

Address: _____

_____

_____

Telephone: _____

The Newsletter is a regular publication of I.P. Sharp Associates. Contributions and comments are welcome and should be addressed to: Jeanne Gershater, I.P. Sharp Newsletter, 145 King Street West, Toronto, Canada M5H 1J8.

Jeanne Gershater, *Editor*
Ginger Kahn, *Assistant Editor*
Ginger Kahn, *Circulation*

Printed in Canada
May 1981
ISSN 0226 854X

I.P. Sharp Associates **Head Office:** 145 King Street West, Toronto, Canada M5H 1J8 (416) 364-5361

International Offices

Aberdeen
I.P. Sharp Associates Limited
5 Bon Accord Crescent
Aberdeen AB 12DH
Scotland
(0224) 25298

Amsterdam
Intersystems B.V.
Kabelweg 47
1014 BA Amsterdam
The Netherlands
(020) 86 80 11
Telex: 18795 ITS NL

Atlanta
I.P. Sharp Associates, Inc.
1210 S. Omni International
Atlanta, Georgia 30335
(404) 586-9600

Boston
I.P. Sharp Associates, Inc.
Suite 415
148 State Street
Boston, Massachusetts 02109
(617) 523-2506

Brussels
I.P. Sharp Europe S.A.
Avenue du General de
Gaulle, 39
1050 Bruxelles
Belgique
(02) 649 99 77

Calgary
I.P. Sharp Associates Limited
Suite 2660, Scotia Centre
700-2nd Street S.W.
Calgary, Alberta T2P 2W2
(403) 265-7730

Canberra
I.P. Sharp Associates Limited
16 National Circuit
Barton, ACT 2600
Australia
(062) 73-3700

Chicago
I.P. Sharp Associates, Inc.
2 North Riverside Plaza
Suite 1736
Chicago, Illinois 60606
(312) 648-1730

Cleveland
I.P. Sharp Associates, Inc.
(216) 431-6861
(local call, switched through
to Rochester office.)

Copenhagen
I.P. Sharp ApS
Ostergade 24B
1100 Copenhagen K
Denmark
(01) 11 24 34

Coventry
I.P. Sharp Associates Limited
7th Floor B Block
Coventry Point, Market Way
Coventry, England CV1 1EA
(0203) 21486/7

Dallas
I.P. Sharp Associates, Inc.
Suite 1148, Campbell Centre
8350 Northcentral Expressway
Dallas, Texas 75206
(214) 369-1131

Denver
I.P. Sharp Associates, Inc.
Suite 416
5680 South Syracuse Circle
Englewood, Colorado 80111
(303) 741-4404

Dublin
Gamma Data Systems Limited
(Agent)
Dollard House
Wellington Quay
Dublin 2, Ireland
(01) 711 877

Düsseldorf
I.P. Sharp GmbH
Leostrasse 62A
4000 Düsseldorf 11
West Germany
(0211) 57 50 16

Edmonton
I.P. Sharp Associates Limited
Suite 2358, Principal Plaza
10303 Jasper Avenue
Edmonton, Alberta T5J 3N6
(403) 428-6744

Gloucester
I.P. Sharp Associates Limited
29 Northgate Street
Gloucester, England GL1 2AN
(0452) 28106

Hamilton
I.P. Sharp Associates Limited
14 Hess South, Hess Village
Hamilton, Ontario L8P 3M9
(416) 527-3801

Houston
I.P. Sharp Associates, Inc.
Suite 375, One Corporate Square
2600 Southwest Freeway
Houston, Texas 77098
(713) 526-5275

Kitchener/Waterloo
I.P. Sharp Associates Limited
3 Menno St.
Waterloo, Ont. N2L 2A4
(519) 884-5420

London, Canada
I.P. Sharp Associates Limited
Suite 510, 220 Dundas Street
London, Ontario N6A 1H3
(519) 434-2426

London, England
(European Headquarters)
I.P. Sharp Associates Limited
132 Buckingham Palace Road
London SW1W 9SA
England
(01) 730-0361
Telex: 8954178 SHARP G

Los Angeles
I.P. Sharp Associates, Inc.
Suite 1230, 1801 Century Pk. E
Los Angeles, Ca. 90067
(213) 277-3878

Madrid
I.P. Sharp Assoc. Ltd.
Serrano 23, Piso 8
Madrid - 1, Spain
(91) 276 70 54

Manchester
I.P. Sharp Associates Limited
Paul House
89-91 Buttermarket Street
Warrington, Cheshire
England WA1 2NL
(0925) 50413/4

Melbourne
I.P. Sharp Associates Pty. Ltd.
520 Collins St., 13th Floor
Melbourne 3000
Victoria, Australia
(03) 614-1766

Mexico City
Teleinformatica de Mexico S.A.
(Agent)
Mail to:
Arenal N 40, Chimalistac
Mexico 20 D.F., Mexico
(905) 550-8033

Miami
I.P. Sharp Associates, Inc.
Suite D, Kennedy Building
14560 N.W. 60th Avenue
Miami Lakes, Florida 33014
(305) 556-0577

Milan
I.P. Sharp Srl (agent I.S.I.)
Via Eustachi 11
20129 Milan, Italy
(02) 271-6541

Montreal
I.P. Sharp Associates Limited
Suite 1610
555 Dorchester Boulevard W.
Montreal, Quebec H2Z 1B1
(514) 866-4981

New York City
I.P. Sharp Associates, Inc.
Suite 210
230 Park Avenue
New York, N.Y. 10166
(212) 557-1200

Newport Beach
I.P. Sharp Associates, Inc.
Suite 1135
610 Newport Center Drive
Newport Beach, Ca. 92660
(714) 644-5112

Oslo
I.P. Sharp A/S
Dronningens gate 34
Mail to: P.Boks 486 Sentrum
OSLO 1, Norway
(02) 41 17 04

Ottawa
I.P. Sharp Associates Limited
Suite 600, 265 Carling Ave.
Ottawa, Ontario K1S 2E1
(613) 236-9942

Palo Alto
I.P. Sharp Associates, Inc.
Suite 201, 220 California Ave.
Palo Alto, Ca. 94306
(415) 327-1700

Paris
I.P. Sharp Sarl.
Tour Neptune, Cedex 20
20 Place de Seine
92086 Paris-la-defense
France
(1) 773 57 77

Philadelphia
I.P. Sharp Associates, Inc.
Suite 604, 437 Chestnut St.
Philadelphia, Pa. 19106
(215) 925-8010

Phoenix
I.P. Sharp Associates, Inc.
Suite 503
3033 N. Central Avenue
Phoenix, Arizona 85012
(602) 264-6819

Rochester
(United States Headquarters)
I.P. Sharp Associates, Inc.
1200 First Federal Plaza
Rochester, N.Y. 14614
(716) 546-7270
Telex: 0097 8473
0097 8474

San Francisco
I.P. Sharp Associates, Inc.
Suite C-415, 900 North Point St.
San Francisco, Ca 94109
(415) 673-4930

San Jose
I.P. Sharp Associates, Inc.
3028A Scott Boulevard
Santa Clara, California 95050
(408) 727-9446

Saskatoon
I.P. Sharp Associates Limited
Suite 208, 135 21st Street E.
Saskatoon, Sask. S7K 0B4
(306) 664-4480

Seattle
I.P. Sharp Associates, Inc.
Suite 217
Executive Plaza East
12835 Bellevue-Redmond Rd.
Bellevue, Washington 98005
(206) 453-1661

Singapore
I.P. Sharp Associates Pte. Ltd.
Suite 1501, CPF Building
79 Robinson Rd.
Singapore 0106
Singapore
223-0221

Stockholm
I.P. Sharp AB
Kungsgatan 65
S111 22 Stockholm, Sweden
(08) 21 10 19

Stuttgart/Boeblingen
I.P. Sharp GmbH
Schafgasse 3
7030 Boeblingen
West Germany
(070 31) 2 30 14

Sydney
I.P. Sharp Associates Pty. Ltd.
Suite 1351, 175 Pitt Street
Sydney, N.S.W., Australia 2000
(02) 232-6366
Freight to: c/-Greenaways
Customs Services
Alexandria, N.S.W.

Toronto
(International Headquarters)
I.P. Sharp Associates Limited
145 King Street West
Toronto, Ontario M5H 1J8
(416) 364-5361

Toronto (Special Systems Div.)
I.P. Sharp Associates Limited
156 Front Street W., 5th Floor
Toronto, Ontario M5J 1G6
(416) 364-5361

Vancouver
I.P. Sharp Associates Limited
Suite 902, 700 West Pender St.
Vancouver, B.C. V6C 1G8
(604) 687-8991

Victoria
I.P. Sharp Associates Limited
Chancery Court
1218 Langley Street
Victoria, B.C. V8W 1W2
(604) 388-6365

Vienna
I.P. Sharp Ges.mbH
Rechte Wienzeile 5/3
A-1040 Wien, Austria
(0222) 57 65 71

Washington
I.P. Sharp Associates, Inc.
Suite 400, 1835 K Street N.W.
Washington, D.C. 20006
(202) 293-2915

White Plains
I.P. Sharp Associates, Inc.
180 East Post Road, LL-6
White Plains, New York 10601
(914) 328-8520

Winnipeg
I.P. Sharp Associates Limited
Suite 208
213 Notre Dame Avenue
Winnipeg, Manitoba R3B 1N3
(204) 947-1241

Zurich
I.P. Sharp A.G.
Fortunagasse 15
8001 Zurich
Switzerland
(01) 211 84 24

SHARP APL Communications Network

APL OPERATOR VOICE (416) 363-2051 **COMMUNICATIONS (416) 363-1832**

Local dial access is available in all locations listed above and in:

• Alliance • Ann Arbor • Austin • Baltimore • Birmingham • Buffalo • Clewiston (Fl) • Dayton • Des Moines • Des Plaines
• Detroit • Ft. Lauderdale • Greenwich (Ct) • Halifax • Hartford • Hull • Knoxville • Laurel • Liverpool • Lyndhurst • Minneapolis
• New Orleans • Oxford • Quebec City • Raleigh • Red Deer • Regina • Santa Ana • Sunnyvale
• Syracuse • Towanda • Ukiah • Warrington

Our private, packet-switched network connects with the Value Added Networks in:

• Alaska • Argentina • Bahrain • Bermuda • Finland • Hawaii • Hong Kong • Israel • Luxembourg • Mexico • New Zealand
• The Philippines • Portugal • Puerto Rico • Spain • Taiwan

In the continental United States the SHARP APL Network is interconnected with the Value Added networks to provide access in 170 more cities, and in Canada with 40 more. In all, with the 80 cities served by the I.P. Sharp Network listed above, SHARP APL is accessible from close to 300 places via a local phone call. Please ask at your nearest I.P. Sharp office for a complete list of access points and access procedures.