Volume 1 No. 5 January/February 1978

Journal
of the HP General Systems Users Group

Executive Board

Inside this issue

Published By-Monthly

Contributions: Address the Journal Editor

Journal Editor Elias Zabor
HP General Systems Users Group, c/o Hewlett-Packard Company, 5303 Stevens Creek Blvd., Santa Clara, CA 95050, (408) 249-7020

Featured articles

Recursive Programming In FORTRAN

by Lonny B. Winrich
Computer Science Department
University of Wisconsin-La Crosse
La Crosse, Wisconsin 54601

1. Introduction

For the adventurous FORTRAN programmer who might like to try a recursive subroutine for implementation of a new algorithm, the literature of computing is discouraging. Most books which treat advanced topics in programming, like Wirth (Ref. 6, p. 128) and Goodman and Hedetniemi (Ref. 2, p. 138), simply state that recursion is impossible in FORTRAN. For many FORTRAN compilers, they are, of course, correct. Even on those systems which permit recursive programs in other high level languages, the technique is usually denied to the FORTRAN programmer.

The sad part is that there seems to be little reason for not permitting recursion in FORTRAN. Stack oriented machine architectures permit relatively natural implementation of subroutine linkage structures which allow recursive techniques. The HP 3000 has such an architecture and it was both encouraging and challenging to find the statement, "A subroutine is reentrant, that is, it can contain a CALL statement which references, or calls, itself either directly or indirectly." in the FORTRAN reference manual (Ref. 3, p. 11-4). The encouragement was short lived however, for although several sections of the System Reference Manual (Ref. 4, p. 4-26ff) are devoted to a discussion of how to implement recursion in SPL, one searches in vain for an example of recursion in FORTRAN. Nevertheless, the challenge remains.

2. An Elementary Example

Since examples of recursive programs in FORTRAN are nonexistent, it seems reasonable to look to examples in other languages and adapt them to FORTRAN. If that is done, for example in Tucker (Ref. 5, p. 50 or p. 262), one soon learns that the first recursive program to be implemented in any language is the familiar factorial function of mathematics. Or so it seems from empirical evidence. The factorial function is conveniently defined recursively as:

$$0! = 1$$
$$N! = N(N-1)! \text{ for } N \geq 1.$$

Since its rendition into FORTRAN seems obligatory, the following program is offered as an elementary example of recursive programming.

```
      INTEGER FUNCTION FACT(K)
      IF(K .LE. 1) GO TO 10
      FACT = K*FACT(K - 1)
      RETURN
   10 FACT = 1
      RETURN
      END
```

A simple main program which invokes this function for a few values of K will soon convince anyone that it works, provided the program is run under a system which permits recursion in FORTRAN such as the HP 3000 Series II. Encouraged by this result, one naturally wants to apply recursion to the implementation of a more useful algorithm. After all, the occasions for computing factorials are relatively limited. There is also some question as to whether a function such as the factorial should be computed recursively (Ref. 6, p. 128) because of inefficiencies.

3. Preorder Traversal of a Binary Tree

Some of the most important and useful examples of recursion in programming come from the use of data structures which are defined recursively. One such data structure is the binary tree, often used to assure efficient searching of an ordered list. The use of this example is motivated by history as well as by logic, for the impetus for this paper came from a discussion in a Data Structures class which the author was teaching at the University of Wisconsin-La Crosse.

Throughout the rest of this paper, the binary tree shown in Figure 1 will be used for examples. Its structure is intended to be representative albeit unre-alisically brief; note that there is at least one root node with a missing left subtree and one with a missing right subtree.

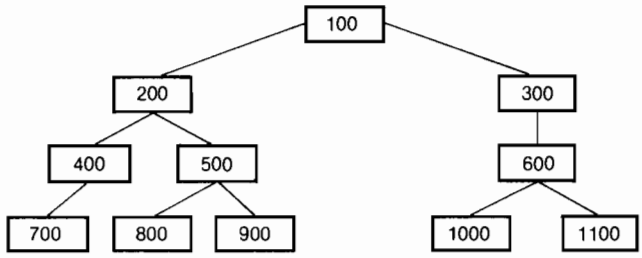

Figure 1

The preorder traversal of such a tree is defined by Elson (Ref. 1, p. 87) to consist of three steps:

a. Process the root node.

b. Traverse the left subtree of the root.

c. Traverse the right subtree of the root.

Although he offers an iterative algorithm for preorder traversal (Ref. 1, p. 91) which does not require recursion, Elson recognizes that the definition of the process is recursive because the second and third steps involve traversals of subtrees of the root of the tree being traversed. Others offer recursive algorithms for the preorder traversal but, of course, not in FORTRAN.

The first step in implementing the preorder traversal in FORTRAN is to render the binary tree of Figure 1 as a FORTRAN data structure. This can be done through the use of a two-dimensional array: ITREE(11,3). The first subscript will serve to identify a particular node of the tree and the second subscript will indicate the data element at that node, the left subtree link and the right subtree link, respectively. Figure 2 gives the structure of this array. – 1 is used as a null pointer in this implementation.

1st Subscript \ 2nd Subscript	1	2	3
1	100	2	3
2	200	4	5
3	300	-1	6
4	400	7	-1
5	500	8	9
6	600	10	11
7	700	-1	-1
8	800	-1	-1
9	900	-1	-1
10	1000	-1	-1
11	1100	-1	-1

Figure 2

Followng Wirth (Ref. 6, p. 199), one is tempted to implement a recursive preorder traversal in FORTRAN with a program similar to the following:

```
 1        SUBROUTINE PREORDER(I)
 2        COMMON ITREE(11,3)
 3        IF(I .GT. 0) GO TO 10
 4        RETURN
 5     10 DISPLAY "NODE = ", ITREE(I,1)
 6        I = ITREE(I,2)
 7        CALL PREORDER(I)
 8        I = ITREE(I,3)
 9        CALL PREORDER(I)
10        RETURN
11        END
```

Program 1

The basic logical structure is no more complicated than Elson's definition of preorder traversal. The variable I is a pointer to the node of the tree which is to be processed. The pointer is checked to determine that it is not a null pointer (line 3) and the node processed, or in this example, simply displayed (line 5). If the pointer is null, the subroutine returns to the calling program. After processing, the pointer is first set to link to the left subtree of the node and the subroutine calls itself (line 7). After processing the left subtree, the pointer is set to the right subtree and again the subroutine recurses (line 9). The structure of the program is delightfully simple and straightforward.

To test this subroutine, a main program is needed. In this experiment the following main program was used.

```
      COMMON ITREE(11,3)
      DO 10 I = 1,11
      ACCEPT (ITREE(I,J),J = 1,3)
   10 CONTINUE
      I = 1
      CALL PREORDER(I)
      STOP
      END
```

The array ITREE was filled as shown in Figure 2 and the result of executing the program was the following printout.

```
NODE =  100
NODE =  200
NODE =  400
NODE =  700
```

The printout is correct as far as it goes. Inserting DISPLAY statements in the subroutine and the main program indicates that the values of the node pointer follow the pattern shown in Figure 3.

	I \ Depth of Recursion	1	2	3	4	5
	1	1				
	2		2			
	3			4		
Number of	4				7	
Linkages	5					-1
	6				-1	
	7			-1		
	8		-1			
	9	-1				

Figure 3

In this table, linkages 1 through 5 are calls to the subroutine and linkages 6 through 9 are returns. The values of I are obtained from execution of the program. The problem seems to arise at the eighth linkage. At this point the node pointer should identify the right subtree of node 2 and recurse with a value of 5. An explanation is clearly in order.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Since subroutine linkages are handled through use of the stack in HP 3000 FORTRAN it would seem that the appropriate values of the argument I should be preserved as the subroutine recurses. The catch is that FORTRAN argument transfers are by location, not by value.

The address of I is transfered to the stack with each call to PREORDER. Since that address comes from the main program, it is the same for each call and the arguments are not preserved for return linkages. An obvious solution is to transfer arguments by value rather than location. Unfortunately the HP 3000 FORTRAN compiler does not allow transfer by value.

To simulate the transfer by value, a programmer-defined stack can be used in common storage. The stack must be managed by the subroutine to transfer the arguments. Program 2 is a version of such a subroutine.

```
1             SUBROUTINE PREORDER
2             COMMON ITREE (11,3), ISTK (10), K
3             I = ISTK (K)
4             IF (I .GT. 0) GO TO 10
5             K = K-1
6             RETURN
7           10 DISPLAY "NODE = ", ITREE(I,1)
8             K = K+1
9             ISTK (K) = ITREE (I,2)
10            CALL PREORDER
11            I = ISTK (K)
12            K = K+1
13            ISTK (K) = ITREE (I,3)
14            CALL PREORDER
15            K = K-1
16            RETURN
17            END
```

Program 2

Except for the added lines associated with stacking and unstacking arguments, the logic of this program is identical to that of Program 1. I is still a pointer to the current node of the tree but it is now a local variable in the subroutine; K is the stack pointer.

With appropriate changes in the driving program, this subroutine traverses the binary tree and generates a correct listing of the nodes. The result seems to lend credence to the explanation based on transfer by location but it must be recalled that no such difficulty was encountered with the factorial function. Why?

An examination of the factorial program indicates that the function recurses by calling itself but the argument involved is an expression. The expression is apparently computed and stored locally within the subroutine before recursion and a unique local address is transfered. The programmer generated stack and the attendant manipulation needed to simulate transfer by value can be avoided if one is careful to use local arguments in recursive calls. Program 3 provides another correct version of the preorder traversal in FORTRAN. This version works with the original main program written for Program 1.

```
1             SUBROUTINE PREORDER (I)
2             COMMON ITREE (11,3)
3             IF (I .GT. 0) GO TO 10
4             RETURN
5           10 DISPLAY "NODE = ", ITREE (I,1)
6             CALL PREORDER (ITREE (I,2))
7             CALL PREORDER (ITREE (I,3))
8             RETURN
9             END
```

Program 3

4. Conclusion

Recursive programming is indeed possible in FORTRAN. All that is required is a compiler such as HP 3000 FORTRAN which handles subroutine linkages in an appropriate fashion. To all the usual caveats about the possible inefficiency of recursive subroutine linkages, another must be added when working in FORTRAN. The FORTRAN programmer must pay close attention to the methods used to transfer arguments to subroutines. If transfer by value is allowed, it would seem to solve the problem. Where it is not, transfer by value can be simulated by use of common blocks. Alternatively, recursive calls can be written in such a way as to use only arguments which are local to the subroutine.

In retrospect it seems strange that the HP 3000 FORTRAN compiler does not allow arguments to be transfered by value. It is one of the few compilers extant which allows recursion in FORTRAN and transfer by value would be useful for such applications. There are several FORTRAN compilers which allow transfer by value but forbid recursion. This seems to be an appropriate place to suggest an additional feature for the HP 3000 FORTRAN compiler.

REFERENCES

1. Elson, Mark, *Data Structures*. Science Research Associates, Inc. Palo Alto. 1975.
2. Goodman, S.E. and Hedetniemi. S.T., *Introduction to the Design and Analysis of Algorithms*. McGraw-Hill Bok Company. New York. 1977.
3. *HP 3000 Series II FORTRAN Reference Manual.* Hewlett-Packard. Santa Clara. 1976.
4. *HP 3000 Series II System Reference Manual.* Hewlett-Packard. Santa Clara. 1976.
5. Tucker, Allen B. Jr., *Programming Languages*. McGraw-Hill Book Company. New York. 1977.
6. Wirth, Niklaus, *Algorithms + Data Structures = Programs*. Prentice-Hall, Inc. Englewood Cliffs. 1976.

● ● ●

PASCAL For The HP 3000

by John Earls
Arthur A. Collins, Inc.
13601 Preston Road
Dallas, Texas 75240

PASCAL is an ALGOL-like programming language defined in 1968 by Niklaus Wirth and others. The first compiler was operational in 1970. The PASCAL User Manual and Report, 2nd edition, by Kathleen Jensen and Niklaus Wirth (Publisher: Springer-Verlag) defines the "standard" PASCAL.

PASCAL is like SPL / 3000 in many ways, but differs in three main areas:

- Machine Independent
- Includes I/O Syntax
- Has extensive data structures

The first area has allowed the language to be implemented on various machines, from Intel 8080's to the Clay-I.

The second area tends to make I/O operate like unformatted FORTRAN I/O; e.g., opens, closes, etc., are done by runtime support.

The third area is illustrated by the following example program.

```
program setop (output):
  type days = (m,t,w,th,fr,sa,su);
    week = set of days;
  var wk,work,free : week;
    d : days;
  Procedure check(s : week);
    var d : days;
  begin write('');
    for d : = m to su do
      if d in s then write('x') else write('0');
    writeln
  end; (*check*)

  begin work : = [ ]; free : = [ ];
    wk : = [m..su];
    d : = sa; free : = [d] + free + [su];
    check(free);
    work : = wk - free; check(work);
    if free <= wk then write('o');
    if wk >= work then write ('k');
    if not (work >= free) then write ('jack');
    if [sa] <= work then write ('forget it');
    writeln
  end.
```

As can be seen, this program (which is complete except for control statements) defines "days" as m thru su and defines a week as a set of days. Then the program processes days directly without resorting to programmer-defined mapping into bits or character strings.

Much of the power of PASCAL is derived from its simple syntax and its ability to define data structure in the terms of the problem at hand.

This short example will not give a real feel for PASCAL. Instead, one should read the book by Jensen and Wirth.

In addition, a very active PASCAL User's Group exists. The all-purpose coupon at the end of this article can be copied and sent with $4 to get you on the mailing list (for each academic year ending June 30th) for the PASCAL News which usually contains about 100 pages of good information about the current state of PASCAL.

There are at least two PASCAL compilers for the HP 3000. At least one will be made available via the Contributed Library shortly, if current plans continue.

PASCAL should be well suited to the HP 3000 because the language is designed around stack architecture. With some compiler effort, detail documentation and a little microcode help, PASCAL on the HP 3000 can be a very powerful tool.

Editor's Note: Readers wishing to contribute an article or opinion as to usage of PASCAL at their installations may submit the contribution to the *Journal*. Additionally, the PASCAL User's Group would welcome such an article; please note PUG's address.

PASCAL User's Group
c/o Andy Mickel
University Computer Center: 277 Ex
208 SE Union Street
University of Minnesota
Minneapolis, MN 55455, USA

☐ Please enter me as a new member of PASCAL User's Group for _____ Academic year(s) ending June 30,_____ .I shall receive all 4 issues of *PASCAL News* for each year. Enclosed please find _____ ($4.00 for each year). When joining from overseas, check the *PASCAL News* POLICY for a PUG "regional representative".

☐ Enclosed please find a contribution which I wish to submnit for publication in the next issue of *PASCAL News*.

From: Name _____

Mailing Address: _____

_____

Phone: _____ Date: _____

Computer System(s) _____

● ● ●

Tips & Techniques

1000th HP 3000 Series II: Central Node In A Distributed Processing Network.

by Journal Editor

General Mills Consumer Food Group of General Mills, Inc., purchased a fifth HP 3000 Series II System; it will be the central node of a distributed processing network designed for production management and inventory control, and will eventually link every major Consumer Food Group Plant. The other four computers are used for order entry, program development, procurement and accounting, and meeting scheduling tasks.

The network will be implemented using DS/3000, HP's networking product for linking HP 3000s over communication lines.

• • •

Treasure In The Contributed Library

by John Earls
Arthur A. Collings, Inc.
13601 Preston Road
Dallas, Texas 75240

The HP 3000 Users Group Library has been a very big dividend on our Users Group membership.

We currently use about a dozen programs from the Library — galley most often. We are a small organization doing R&D (4 secretaries and about 16 staff members). Everyone has a 2645 terminal. All typing is input directly to the computer via Editor. Diablo 1620's are used for final output. This has provided us with the ability to produce high quality revised papers on very short notice.

We had very little problem with any of the programs from the library. Our approach to using a program is as follows:

(1) GETFILE (a program on the User Library) is used to selectively bring into a convenient account.
(2) The documentation is listed and read in detail.
(3) File opens and closes are examined in the source.Adjustments are made to suit our operation.
(4) The program is recompiled and stored in our User Program Group (xxx.P.SYS).
(5) Documentation is made available to potential users.

The only time we have had problems occurred when we attempted to use programs directly. Then, all of those system parameter choices we made for various reasons bite back.

We started using simple programs from the library. Now with the above procedure, complex programs such as "ECAP" and "PERT" have been made operational in a few hours.

• • •

COBOL Reserved Words

by Greg Gloss
HP General Systems Division
5303 Stevens Creek Blvd.
Santa Clara CA 95050

At a recent Regional Users Group meeting, a request was made for a list of words which are not currently on the COBOL/3000 Reserved Words list, but which may be included in the future. The following two lists show those words which are not currently reserved in COBOL/3000 but which are on the ANSI-COBOL, X3.23-1974 Reserved Word list and those words which are tentatively planned for extensions to COBOL/3000 at a future date.

The following Reserved Words have been added to ANSI-COBOL in the 1974 standard:

ALSO
BOTTOM
CANCEL
CD
CHARACTER
CODE-SET
COLLATING
COMMUNICATION
COUNT
DATE
DAY
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DELETE
DELIMITED
DELIMITER
DESTINATION
DISABLE
DUPLICATES
DYNAMIC
EGI
EMI
ENABLE
END-OF-PAGE
EOP
ESI
EXCEPTION
EXTEND
INITIAL
INSPECT
LENGTH
LINAGE
LINAGE-COUNTER
MERGE
MESSAGE
NATIVE
ORGANIZATION
OVERFLOW
POINTER
PRINTING
PROCEDURES
QUEUE
RECEIVE
REFERENCES
RELATIVE
REMOVAL
REWRITE
SEGMENT
SEND
SEPARATE
SEQUENCE
SORT-MERGE
STANDARD-1
START
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUPPRESS
SYMBOLIC
TABLE
TERMINAL
TEXT
TIME
TRAILING
UNSTRING

The following words are tentatively planned for HP extensions to COBOL at a future date. This list is subject to change.

C01	INTRINSIC
C02	LABELS
C03	NOLIST
C04	SEQ
C05	SW0
C06	SW1
C07	SW2
C08	SW3
C09	SW4
C10	SW5
C11	SW6
C12	SW7
CC	SW8
CONDITIONALLY	SW9
EBCDIC	UN-EXCLUSIVE
EXCLUSIVE	VOL
EXDATE	WHEN-COMPILED
FREE	

• • •

Need A Little More Data Stack Space?
Try NOCB On Your RUN Statement

by Jack Howard, HP Neely Sales Region
Computer Systems
Systems Support Representative
P.O. Box 92105
Los Angeles CA 90009

NOCB on your RUN statement directs MPE not to put your PCBX on your data stack. This increases your usable data stack space by 258 to 768 bytes.

Note that the PCBX will be kept in an extra data segment; therefore, execution time will increase, but this increase is usually relatively small.

Contributed Library Corner

Contributed Library To Be Enhanced

by Wayne Holt
Whitman College
Walla Walla WA 99362

The Library Committee will immediately begin a Library design and restructure project, the completion of which will be timed to coincide with the release of MPE II-B. Because of this, the next Library release will probably be delayed until May or June. Thereafter, it should follow its normal cycle. The new Library will incorporate at least two new items, and probably more. First, a comprehensive data base will exist containing all relevant data about Library contributions, the clearinghouse, and all group publications. It will initially be structured in an IMAGE base, allowing users to run QUERY against it. In addition, a sequential print file will be available for those sites without IMAGE/QUERY. Eventually, a KSAM base may also be implemented.

Second, a grading system will be implemented and applied to software in the Library and reported in the Library data base. The following has been suggested as a possible scheme:

A — Performs as documented under a specified MIT. This grade indicates that positive user responses have been received or that a quality review has been conducted.

B — Execution with sample data has been successful under a specified MIT. This grade indicates that someone involved with the User Group Library has successfully executed the program using sample data provided by the contributor. No other quality reports are available.

C — No quality statement is available. This grade indicates that the program has not been submitted to any quality checks and no user reports are available. Initially, all new contributions (and all of the present library) would start here.

D — Does not execute properly under a specified MIT. This grade indicates difficulties with a program due to an MIT update or for other reasons.

Your help is needed to make this grading system work. If your site would be willing to *occasionally* review new contributions within a Library Class group, please contact me as soon as possible:

Wayne Holt
Box F-69
Whitman College
Walla Walla WA
99362
(509) 527-5417

LISA — An Interactive Statistical Program for the HP 3000

by James P. Schwar
Computer Center Director
Lafayette College
Easton PA 18042

Introduction

Lafayette Interactive Statistical Analysis (LISA/3000) is an interactive program designed for the statistical analysis of a data matrix. Data handling as well as statistical procedures are an integral part of the program and over 30 such programs are available. The source language is HP 3000 FORTRAN coupled with calls to system intrinsics. The data matrix is copied from a user file and saved for program use as a temporary disk file. Source code for version .03 is available from the contributed library.

Data Matrix

The data matrix resides in a work area created by the programmatically issued BUILD LISA15;DISC = 2000,10,1;REC = 64;TEMP command. An identical scratch file LISA16 is also built. Both files are purged prior to exiting LISA. The data matrix is limited to 32 variables (maximum) by 2000 observations (maximum). A greater number of observations can be utilized if the BUILD commands are issued prior to running the program. The number of observations, for the external BUILD command, is limited to 32767.

The original data matrix is a user created ASCII file. LISA issues the file equation FILE FTN13 = filereference, OLD where filereference is the name of the user created ASCII file. Normally, filereference will be an editor file where each line or record is an observation. Input is in free form and each data item must be separated from its neighbor by one or more blanks. The first read from FTN13 scans record one and sets the number of variables while the EOF terminates data input and sets the number of observations. The original data matrix has now been copied to LISA15 and FTN13 is reset and closed. Unused columns are zeroed during the write to LISA15. The temporary disk area beyond the row limit is unused but available for future expansion of the data matrix. LISA16 is a scratch file required by several of the statistical procedures.

LISA performs data handling and statistical analysis by accessing user requested procedures. Each procedure has a unique identifier and the program prompts for this identifier by asking

WHICH PROCEDURE?

A procedure will also prompt for data entry.

Data Handling Procedures

The data handling and utility procedures available in LISA are:

APPEND allows the contents of an ASCII file to be appended to the data matrix. The number of variables is reset by the appended file. The number of observations will be the sum of the observations for the old data matrix plus those for the appended data matrix.

DATA replaces the current data matrix with the contents of the ASCII file specified. Column and row limits are also reset.

EDIT allows the data matrix to be edited. Options include add, change, delete, list, replace and verify.

EXIT terminates LISA/3000.

HELP briefly explains the procedures available in LISA/3000.

PRINT displays all or part of the contents of the data matrix.

PURGE will purge the file (name) selected.

RESET will reset the number of variables (column limit) for the data matrix.

SAVE allows the data matrix to be saved as a binary file whose code is 999.

SORT calls the SORT intrinsic thus permitting the entire data matrix to be sorted on the variable selected.

SUBSET permits a starting and ending row to be specified so that a subset of the data matrix can be analyzed.

SYSTEM causes a program break so that MPE commands can be issued.

TRANSFORM allows the selected variable to be transformed into itself or a new variable. Approximately twenty commonly used transformations are available.

TRAP enables the arithmetic traps. These traps are initially enabled.

UNTRAP disables the arithmetic traps.

VERIFY displays the status of the data matrix.

***COMMENT** accepts up to forty characters following the * and prints same as a comment.

***BINARY** allows a saved file to be recalled.

***KEYIN** permits the data matrix to be entered via the terminal keyboard.

***NONE** allows the user to issue the MPE file equation FILE FTN13 = filereference,OLD.

Statistical Procedures

The following statistical procedures are available:

ANOVA performs a one-way analysis of variance on the selected variable.

CANONICAL performs canonical correlation on the data matrix.

CHISQ calculates a contingency table and chi-square for a user generated table of occurrences.

CORRELATE calculates the cross products and correlation coefficients for all variables in the data matrix.

DISCR performs discriminant analysis on the data matrix.

ELEMSTAT calculates for a selected variable or all variables in the data matrix: mean, standard deviation, standard error, variance, skewness, kurtosis, maximum, minimum and the range.

FACTOR performs factor analysis on the data matrix.

FREQUENCY outputs an open-ended frequency table and histogram for the selected variable.

OCCUR analyzes a matrix of occurrences which is generated by tabulating the number of occurrences of an event. This count is generated within the procedure. CHISQ is an entry point of this procedure.

POLYNOMIAL regresses the selected dependent variable Y against the selected independent variable X. The polynomial thus fitted is of the form

$$Y = A0 + A1*X + A2*X**2 + \ldots$$

and may be of degree one through eight.

REGRESSION performs multiple linear regression on the selected variables. One dependent and several independent variables may be specified. Output consists of the regression coefficients, intercept, multiple correlation coefficient, standard error, analysis of variance for the regression and an optional table of residuals.

SCATTER outputs a scatter diagram for the two variables selected.

SELECT will selectively delete rows (observations) from the data matrix for the selected variable. These deletions are specified by a 'FORTRAN-like' logic test.

SMOOTH outputs a table of the triple exponentially smoothed values for the selected variable.

TABULATE allows for the tabulation of the number of occurrences of an event for the selected variable. This tabulation or count uses a 'FORTRAN-like' logic test. Procedure OCCUR calls TABULATE to generate the occurrence matrix.

TREND performs elementary trend analysis on the data matrix. Options thus far implemented are selected a random subset of the data matrix, lag or lead a variable and difference a variable.

Statistical References

Procedures CORRELATE and ELEMSTAT use the equations found on pages 2-5 through 2-11 of the HP3000 Scientific Library Reference Manual (03000-90010). The IBM Scientific Subroutine Library supplied many of the remaining statistical routines. Some modifications to the IBM routines were necessary so as to maintain compatibility with the data matrix. These subprograms and their associated procedures are:

Subprograms	Procedure
COORE,MULTR,ORDER	REGRESSION
COORE,MULTR,ORDER	POLYNOMIAL
TRACE,VARMX,LOAD	FACTOR
DMATX,DISCR	DISCR
CANOR,NROOT	CANONICAL

The remaining procedures were coded specifically for LISA. Real arithmetic is single precision and LISA should run on both the CX and Series II machines. Changing to double precision arithmetic, particularly in the REGRESSION module and matrix inversion, would improve computational accuracy.

Timing Test

Several of the procedures were run against a random data matrix 32 variables by 2000 observations. The approximate central processor time, as returned by the intrinsic PROCTIME, was recorded for each procedure. The following table summarizes these results as run on a dedicated CX machine with 128K memory. The data matrix was resident on a 47M ISS disk.

Procedure	CPU Seconds	Comments
ANOVA	30	1 variable; 4 groups of 500 observations each.
CANONICAL	160	32 variables; 2000 observations
CORRELATE	240	32 variables; 2000 observations
DATA	60	32 variables; 2000 observations
DISCR	500	32 variables; 4 groups of 500 observations each
ELEMSTAT	85	32 variables; 2000 observations
FACTOR	290	32 variables; 2000 observations
FREQUENCY	30	1 variable; 4 intervals; 2000 observations
POLYNOMIAL	110	2 variables; 2000 observations; any degree
REGRESSION	135	2000 observations; any combination of variables
SCATTER	70	any 2 variables
SMOOTH	90	any variable; 2000 observations
SORT	50	any variable (ascending or descending); 2000 observations.

Corrections/Modifications

This latest release of LISA/3000 incorporates:

1. The new procedures
   EDIT SYSTEM TRAP TREND UNTRAP
2. correction to procedure SORT
   An EOF mark is written at the logical end of file so that the file (data matrix) is sorted correctly.
3. LISA formatted files are now saved with a file code of 999.
4. source documentation in the form of comment cards has been expanded.
5. procedure CORRELATE was rewritten to reduce the CPU time required for computation. Only the lower triangular matrix is calculated for cross-products and correlation coefficients.

• • •

The Clearing House

SPSS for the HP3000

SPSS, the Statistical Package for the Social Sciences, is presently in use at over 1,500 installations around the world and is, undoubtedly, the most popular statistical package in existence today.

The official FORTRAN version of SPSS 7.0 has been converted to the HP3000 and is now available from McMaster University.

Among the many statistical options within SPSS are summary statistics, cross-tabulation, frequency distributions, analysis of variance, regression analysis, factor analysis, canonical correlation and non-parametric statistics. Also included within SPSS are many data manipulation and data file management options.

For more information on SPSS for the HP3000, please contact:

Gary D. Anderson,
Dept. of Clinical Epidemiology and Biostatistics
McMaster University,
1200 Main Street West,
Hamilton, Ontario. L8S 4J9
Phone: (416) 525-9140 x 2437

Editor's Note: *SPSSHP developed at DePaul University was reported in the Nov/Dec Journal issue.

• • •

Message Facility

This flexible package provides storage and retrieval of lengthy messages between interactive users on the HP 3000. Going far beyond the MPE capabilities, the system provides these capabilities:

1. Sending messages of any length, and a "CC" ability to copy the message to other users without re-entering.

2. Scanning headers for messages coming in and messages sent out.
3. Retraction of a message by the sender.
4. Defining distribution lists for messages.
5. Sending system-wide messages.
6. Monitoring of messages to determine when they are received.
7. A "tickler" to notify interactive users when there are urgent messages waiting.

The message facility is an extremely effective tool for communication and coordination among users on the system; a similar package was used in the Carter Presidential Campaign. Price $2500. Contact Binary Associates.

Optional APL Module extends the message facility to APL users and allows the sending of TELL and TELLOPS to and from APL users. Price $500.

Optional Conference Module allows interactive conversations between any number of on-line users (including APL users if the APL module is ordered). Price $2000.

Tape Catalog System

Built around an IMAGE data base, this sytem provides users and librarians with easy on-line access to tape file information. It allows access to information by volume, file name, user, group and account specifications, has provisions for handling cyclical tape files such as system back up, and maintains security compatible with the HP-3000 accounting structure. A utility program is also provided to help keep track of dormant and expired files, and aid in the prevention of erroneous scratching of tape files. Price $2500. Contact Binary Associates.

APL Component File System

APL users typically think of data in terms of variables, from scalars to arrays of up to 63 dimensions. Many who have used APL on non-HP systems are accustomed to files which allow reading and writing of whole variables (or "components") at one stroke. Binary Associates now provides a file system for APL which functions in precisely this manner. Variables of any number of dimensions are written to a file with a single statement, without regard to the shape, size or type of the data. The variables are read from the file and/or updated by reference to the component number (the position of the component in the file: 1st, 2nd, 3rd, etc.), allowing the APL user to treat the entire variable as a single record, regardless of the data it may contain. In addition, it is significantly faster than the shared variable files provided with APL/3000. Price $5000. Contact Binary Associates.

APL Report Formatter

This product gives the APL user access to a powerful, easy to use formatter. It is especially suited for tabular report generation, and uses significantly less C.P.U. time than any other APL/3000 format method. The following formats are supported in the basic module:

A	Character formatting
D	Double precision exponential
E	Exponential (scientific) notation
F	Fixed point
G	Fixed point/exponential as necessary
I	Integer
N	Fixed point with commas grouping every 3 digits to the left of the decimal point
M	Same as N with dollar signs
X	Spaces
T	Absolute column location
	Literal strings embedded in formatted output

For a more complete description of these format types, see the HP FORTRAN manual. Basic module price is $3000, with custom formats or declarations built in for a small additional charge. Contact Binary Associates.

APL Workspace Conversion

A package of programs which will convert non-HP workspaces to HP virtual workspaces is now available from Binary Associates. The workspaces are first written to a specially formatted tape on the external system. The HP programs, running in background, read the tape and re-create the workspaces on the HP3000. As many as 50 workspaces can be converted in a single day. Purchase price $20,000. Workspace conversion performed at $40 per workspace, plus machine time, with a minimum charge of $500.

Extensive discounts available on all software and services. Documentation and 1 year of support and update services included in price. Custom modifications available on a fixed price basis. For further information, contact:

Binary Associates
Software Products Division
911 Kenbrook Drive
Silver Spring, Maryland 20902
(301) 649-2367

• • •

QEDIT

Quick Program Editing

Small Appetite for Computer Time

QEDIT is a specialized text editor for HP 3000 programmers. According to its designers, the benefits provided by this new product are:

1. *REDUCED PROGRAMMER LOAD ON THE COMPUTER*

   The computer supports more terminals and/or provides better response time because the programmers are using fewer system resources (especially disc accesses).

   QEDIT provides a compact workfile which can be directly edited and compiled. The workfile thus replaces both the standard HP source file and the EDIT/3000 K-file. By eliminating high-overhead "TEXT" and "KEEP" operations, QEDIT dramatically reduces the elapsed time, CPU time and disc input/output required for development, leaving more computer power for end-users.

   QEDIT can be instructed by the System Manager to restrict certain programmer actions (such as CHANGE string in all) to low-priority subqueues, and the programmer can switch subqueues dynamically.

2. *INCREASED PROGRAMMER PRODUCTIVITY*

   Programmers accomplish more because QEDIT provides fast, simple features tailored expressly for the editing and compiling of source programs in COBOL/RPG/FORTRAN/SPL.

   QEDIT takes into account the designated source program language (COBOL, FORTRAN, RPG or SPL) in all functions. For example, when performing searches for strings in SMART mode, it looks only for occurrences of the target string as a valid identifier.

QEDITS cuts down waiting time. For example, a programmer can switch from editing one file to another in 1 second.

QEDIT is available for immediate delivery under a rental plan. The rental fee is $100/Mo or $960/Yr, and includes maintenance and a discount for pre-payment of 4 to 16 months. QEDIT provides a price/performance improvement even on installations with only two on-line programmers.

For additional details, including a list of commands and QEDIT performance, please write the source. Contact:

Robelle Consulting Ltd.
#130-5421 10th Avenue
Delta, B.C., Canada V4M 3T9
(604) 943-8021

In case of Canadian postal strike:

Robelle Consulting Ltd.
P.O. Box 501
Point Roberts, Washington, USA 98281

• • •

Text Processing

PACKAGE: ALTER, a text processing language. With over 50 commands oriented specifically towards text handling; this procedure can be called from any language. When ALTER is called from the HP EDITOR/3000, it provides superior editing capabilities, and allows a user to construct special commands.

PRICE: $1000 purchase; $120 annual maintenance. Free evaluation trial provided (30 days).

CONTACT: Chuck Villa *or* Ross Scroggs
Alter*Ability
534 Rosal Ave.
Oakland, CA 94610
(415) 835-5603

• • •

Services

SERVICE: Installation training and consulting. Site- and task-specific training of new users. Development of tools and procedures used to optimize system performance. Expertise in data communications and word processing for the HP 3000.

PRICE: Call for quotation. Special rates for OEMs.

CONTACT: Charles J. Villa, Jr.
154 Laidley Street
San Francisco, CA 94131
(415) 282-8139

• • •

Services

SERVICE: Full range of consulting services for both new and established HP 3000 users, and feasibility studies for prospective users.

CONTACT: William Bryden
Inland Systems Engineering
P.O. Box 925
Redlands, CA 92373
(714) 792-0323

● ● ●

SERVICE: Full time-sharing services at competitive rates on our in-house HP 3000.

SERVICE: Contract programming and consulting. Software development at your site or on our in-house HP 3000 II by our experienced staff of professionals.

CONTACT: Collier Jackson and Associates, Inc.
2909 Bay to Bay Blvd., Suite 210
Tampa, FL 33609

● ● ●

Hardware

EQUIPMENT: Two ISS discs for sale. Interested?

CONTACT: Dennis Lamb, Director of Data Processing
Universal Motor Oils
(313) 264-9387

● ● ●

Miscellaneous

HELP! Anyone who has had experience utilizing a UCC 2000 (or Bromall 2000) X-Y plotter with the HP 3000.

CONTACT: Mr. Charles Hodge
Boyle Engineering
P.O. Box 3030
Newport Beach, CA 92663
(714) 752-0505

● ● ●

All About Us

Two Separate Questionnaires With This Issue

Within the envelope that brought you this month's issue of the Journal, you should have received SYSTEM BACKUP QUESTIONNAIRE and CONTRIBUTED LIBRARY QUESTIONNAIRE.

Please provide your best replies and return at the earliest possible convenience — we've even included an envelope: simply add postage and repeat the envelope address.

Participate in creating a more effective users group. We're trying to learn your needs, transmit them to HP, and derive more benefits for you!

● ● ●

Update To Regional User Groups Listing

The listing which appeared in Vol. 1 No. 4 (November/December of 1977, p. 9) is modified as follows:

North America

CCRUG Central Canada (Toronto)	Doug Wilson Conestoga College 299 Doon Valley Drive Kitchener, Ontario Canada N2G 4M4 (519) 653-2511

The group now meets on a quarterly basis, on the second Tuesday of the month, i.e., February, May, August, November.

Europe

Belgium and Holland	No present reference

● ● ●

Users Group Now A Corporation

"This is a necessary and welcome step that takes place in the transition from a club-type operation to a professional business organization", according to William Bryden, Chairman of The Executive Board.

There are many reasons for taking this step, according to Bill, and these include:

- Protection of the funds of the organization. The Executive Board is now in the process of trying to establish the Users Group as a 'not for profit' organization, and is attempting to obtain tax-exempt status. A side benefit of such a designation is that we may then petition the postal authorities for special rate first class postage. When mailing updated contributed library tapes, the conservation of funds would be considerable if such a rating were granted.
- Help clarify our independence from HP as a separate entity.
- Limit the personal liability of individuals acting on behalf of the Users Group.
- Raise the professional level of the organization.

The intent is also to increase services to users. In this respect, the possibility is being studied toward the establishment of an executive secretary and/or a minimal staff — so that User Group commitments can be met on a timely basis.

About The Name Change

You may have already noticed the name change on the cover of the Journal. While undergoing the legalities of incorporation, it was desirable to select a name that would not exclude future users as prospects for membership in the Users Group and to make available the benefits of membership to all. In the past, HP 3000 designated *the* product of the HP General Systems Division. This number will not necessarily be all inclusive of the range of products of the Division in the future. Consequently, we allow for growth as the need becomes evident, and part of that is by establishing a name that will not need to change — hence, HP General Systems Users Group.

Journal of the HP General Systems Users Group

Volume 1 No. 5 January/February 1978