Hewlett-Packard

# System 45B

## Desktop Computer

# General Utility Routines

HP makes no express or implied warranty of any kind, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose, with regard to the program material contained herein. HP shall not be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance or use of this program material.

Use of the manual and tape cartridge (or flexible disk) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form, or with alterations, is expressly prohibited.

Part No. 09845-10201

Printed in U.S.A.

# General Utility Routines

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525

TABLE OF CONTENTS

PAGE

## Introduction to the 9845 Utility Library

The 9845 Utility Library is a collection of general-purpose software which addresses some problems common to many types of users. The Utility Library, besides being useful in a problem-solving capacity, is also useful for familiarizing the user with the 9845 , by demonstrating the machine's interactiveness, and by demonstrating the machine's features and potential. In addition, the 9845 Utility Library provides examples of programming techniques for inexperienced users.

General comments on the use of the 9845 Utility Library:

There are two types of programs on the 9845 : main programs, and subprograms. The FORTRAN - like subprogramming feature of the 9845 allows program modules to be both application-independent and driver-independent. The subprograms in the Utility Library are application-independent because no I/O is done in the subprograms - they are strictly oriented toward computation. Thus, a teacher who wishes to sort a set of test grades can use the same subprogram as a meteorologist who needs to sort a set of wind-velocity readings. Subprograms are driver-independent because all variables in a subprogram are local to that particular subprogram. A variable in a subprogram does not affect a variable of the same name in the main program (unless the variable is passed through the parameter list).

All subprograms in the Utility Library are stored in string form in DATA files. They may be loaded into memory via the "GET" or "LINK" commands. The calling syntax and parameter list of each subprogram is explained in detail in its own section (refer to the Table of Contents). Thus, if the user wants to write his own driver for a subprogram, first he would look in the documentation to learn what the subprogram does, how to call the subprogram, and where the subprogram is stored. Next the user can write his driver. Last, the user must load the subprogram into memory behind his driver.

Example:

Suppose the user wants to use the subprogram Vectorsort_q, which is stored in file "QSORTN." Suppose also that the user's driver occupies lines 10 through 2560 in memory. In order to load the subprogram into memory behind his driver, the user would perform the following instructions:

Type: GET "QSORTN", 2570
Press: EXECUTE

The 9845 would then find the file "QSORTN" and load its contents into memory, starting at line 2570.

All main programs (including drivers that have been provided for the subprograms) are stored in internal form. A main program is loaded into memory via the "LOAD" instruction.

Example: Suppose the user wants to run the loan amortization program in file "LOAN". To do this, he would perform the following instructions:

Type: LOAD "LOAN"
Press: EXECUTE

The 9845 would then find the file "LOAN" and load its contents into memory. To run the program, press RUN.

Note: The "LOAD" instruction erases any other programs in memory.

A few general comments on the operation of the 9845 :

All commands to the system are performed by typing the command and pressing the EXECUTE key (i.e., 2+2 EXECUTE, GET "LOAN:T15", EXECUTE, MASS STORAGE IS ":T14", EXECUTE, etc.)

When a program asks you to enter something from the keyboard, first type in the proper value (either numeric values, or character and string values) and press the CONT key (CONT stands for CONTINUE).

If you are typing in your own program, program lines are entered into memory by pressing the STORE key (example: 10 PRINT "HELLO" STORE).

If you confuse the use of the EXECUTE, CONT, and STORE keys, the 9845 will usually issue an error message to you. Then you may press RECALL to retrieve the last line you typed, and press the correct key. It should be noted that when the program is asking for a numeric input from the keyboard, if the user enters a number and presses EXECUTE instead of CONT, the value he entered will be displayed on the bottom line of the CRT. No error message will be issued, but the prompt will remain in the display line of the CRT, indicating that the program has not received its value yet.

Subprogram Name: Vectorsort_q

This subprogram allows the user to sort a one dimensional array, or any subset of the array, into ascending or descending order.

This sort is faster than the Minimum Storage Sort on page 31 but it takes more memory. If speed is your major objective, use this routine, but if memory space becomes important, it may be necessary to use the Minimal Storage Sort (For more details, refer to the section under Special Considerations and Programming Hints on page 11 .

Subprogram Utilization:

File name: QSORTN
Calling syntax: CALL Vectorsort_q(A(*), I1, J1, Incdec)

Input Parameters:

A(*) - The first parameter in the parameter list must be a one-dimensional full-precision numeric vector. This is the vector which will be sorted.

I1 - The second parameter must be a full-precision variable, constant, or expression. I1 represents the lower bound of the sort.

J1 - The third parameter must be a full-precision variable, constant, or expression. J1 represents the upper bound of the sort.

Incdec- The fourth parameter must be a full-precision variable, constant, or expression. It indicates whether the sort will be ascending or descending.
If Incdec = 0, the sort will be descending.
If Incdec = 1, the sort will be ascending.
If Incdec = anything else, the vector will not be sorted.

Output Parameters:

A(*) - After the subprogram is executed, this vector will be sorted from elements I1 through J1 (if the value passed in Incdec is 0 or 1 (see the section under Input Parameters)).

Local Variables

I - lower subscript of array segment
J - upper subscript of array segment
K - temporary storage for I
L - temporary storage for J
L(*)- stack for lower endpoints of array segments
Log - The dimensions of the arrays L(*) and U(*) ($Log_2$ of the number of elements to be sorted).
M - stack pointer
T - temporary storage for switching elements
T1 - temporary storage for switching elements
U(*)- stack for upper endpoints of array segments.

Special Considerations and Programming Hints:

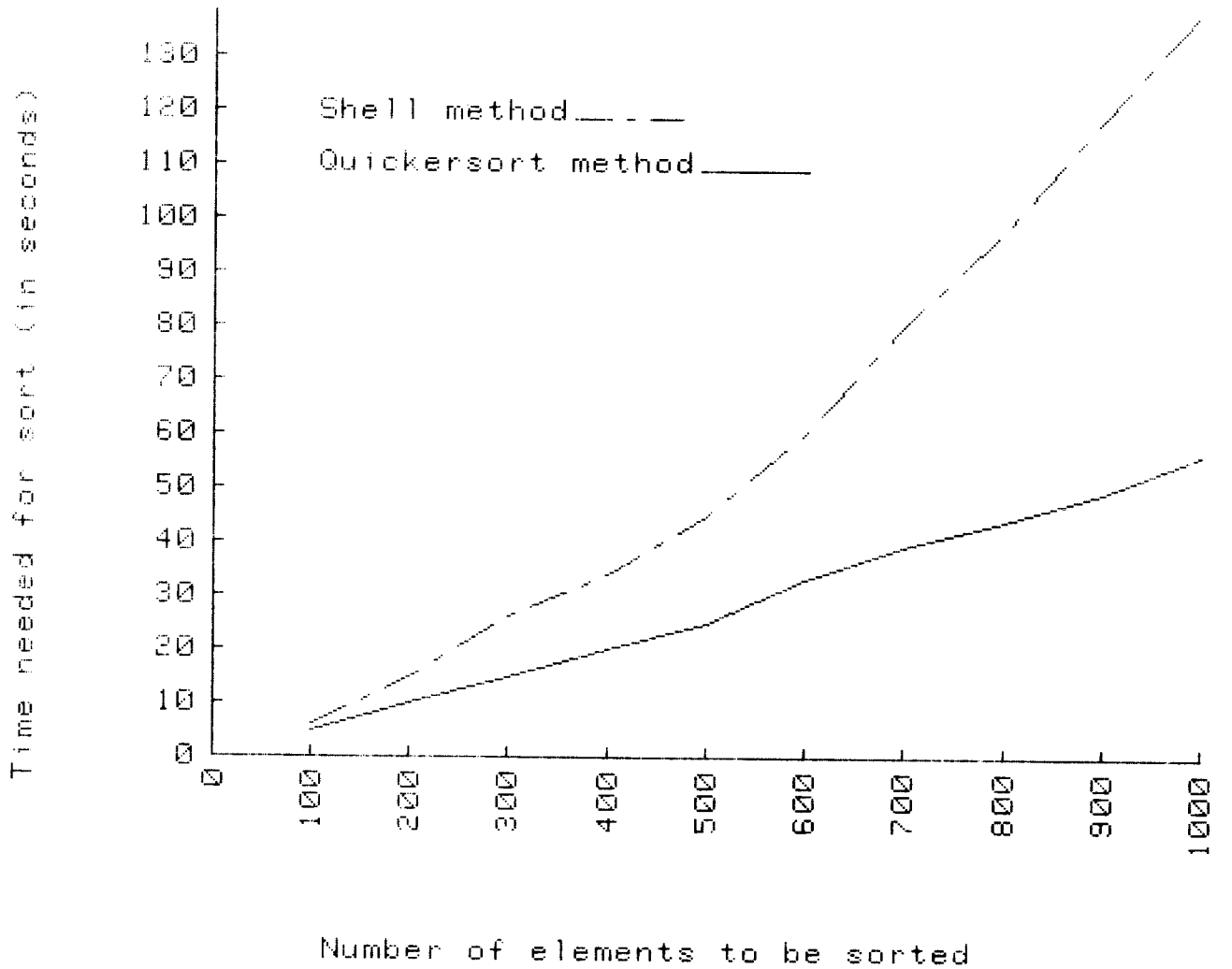
1. If the value passed in for the lower bound of the sort is greater than or equal to the value passed in for the upper bound of the sort, the array will not be changed from its input state.
2. As mentioned in the introduction, this sort takes more memory than the Minimal Storage Sort (Shell sort) on page 31 but it is much faster. (Note: See the timing comparison graph immediately following this section - GRAPH 1.) The subprogram Vectorsort_q needs more bytes of memory than the subprogram Vectorsort_s.
3. This sort may be made faster by removing the capability of sorting both in ascending and descending order. An extra listing is included showing which lines need to be deleted in order to accomplish this. The modified listing (the listing may be found immediately following this section) will sort in ascending order. To sort in descending order, merely reverse the inequality comparisons on all statements having "I" labels (I1, I2, I3, etc.) Thus, statement I1 would become:

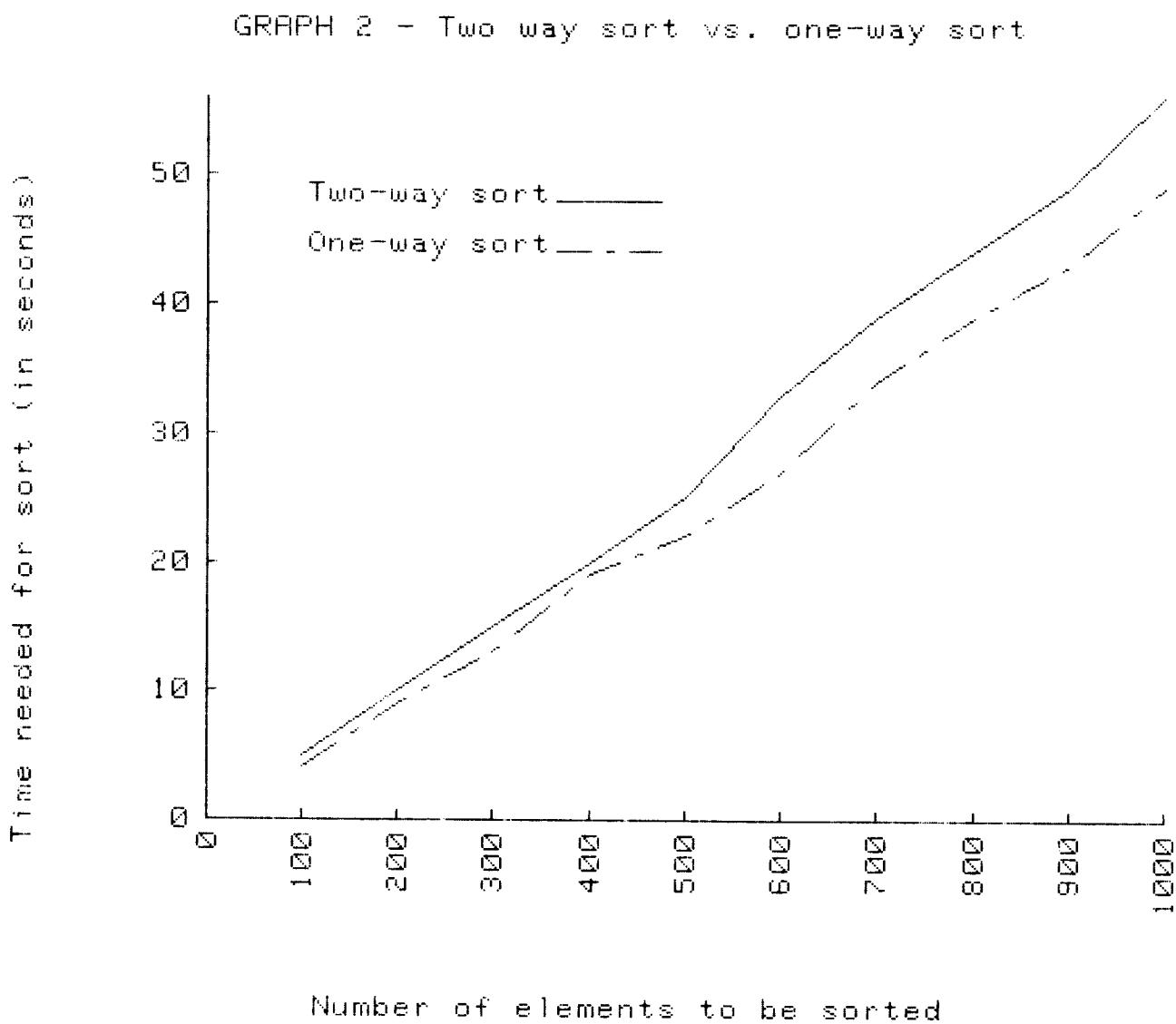
   I1: IF A(I)> = T THEN Lowmiddle1

   GRAPH 2 immediately following this section shows how much improvement in speed is gained by using the procedure outlined above.
4. System Configuration:

   Standard memory option
   (Optional) Printer

GRAPH 1 - Quickersort vs. Shell sort




GRAPH 2 - Two way sort vs. one-way sort
Two-way sort
One-way sort
Time needed for sort (in seconds)
0
10
20
30
40
50
0
100
200
300
400
500
600
700
800
900
1000
Number of elements to be sorted

Modified Subprogram Listing (Ascending sort only)

This listing shows those lines which can be removed to make the sort faster. (The deleted lines would allow the user to sort both in ascending and descending order if they were left in.)

```
890 SUB Vectorsort_q(A(*),I1,J1,Incdec)
900       N=J1+1-I1
910       Logtwo=INT(LGT(N)/LGT(2))+1
920       CALL Qsort(A(*),Logtwo,I1,J1,Incdec)
930 SUBEXIT
940 SUB Qsort(A(*),Log,I1,J1,Incdec)
950  OPTION BASE 1
960  DIM L(Log),U(Log)
970       M=1                                  ! Set stack pointer.
980       I=I1                                 ! Set lower endpoint.
990       J=J1                                 ! Set upper endpoint.
1000 Start1:IF I>=J THEN Nextgroup
1010 Start2:K=1
1020        I2=INT((J+I)/2)                    ! Determine the midpoint of a
 segment.
1030        T=A(I2)
1040       IF Incdec=0 THEN D1
1050 I1:    IF A(I)<=T THEN Lowmiddle1
1060       GOTO 1080
1072 D1:    IF A(I)>=T THEN Lowmiddle1         ! Check to see if lower endpo
int and
1080        A(I2)=A(I)                         !  midpoint are in order.  If
 not,
1090        A(I)=T                             !  switch them.
1100        T=A(I2)                            ! Reset midpoint.
1110 Lowmiddle1: L=J                             ! Set upper endpoint.
1120             IF Incdec=0 THEN D2
1130 I2:         IF A(J)>=T THEN Middlehigh
1140             GOTO 1160
1150 D2:         IF A(J)<=T THEN Middlehigh     ! Check to see if the mid
point and
1160             A(I2)=A(J)                     !  the upper endpoint are
 in order.
1170             A(J)=T                         ! If not, switch them.
1180             T=A(I2)
1190             IF Incdec=0 THEN D3
1200 I3:         IF A(I)<=T THEN Middlehigh
1210             GOTO 1230
1220 D3:         IF A(I)>=T THEN Middlehigh     ! Check to see if the swi
tch left
1230             A(I2)=A(I)                     !  the lower endpoint and
 the mid-
1240             A(I)=T                         !  point in order.
1250             T=A(I2)                        ! If not, switch them.
1260 Middlehigh: L=L-1                          ! Decrement the upper end
point.
1270            IF Incdec=0 THEN D4
```

```
1280 I4:          IF A(L)>T THEN Middlehigh
1290              GOTO 1310
1300 B4:          IF A(L)<T THEN Middlehigh   ! Check to see if the new
 upper
1310              T1=A(L)                     !  endpoint is in order.
1320 Stepup: K=K+1                      ! If not, save the upper endpoint
ond
1330         IF Incdec=0 THEN I5
1340 I5:      IF A(K)<T THEN Stepup     !  increment the lower endpoint.
Now
1350         GOTO 1370
1360 B5:      IF A(K)>T THEN Stepup
1370          IF K>L THEN Passed        !  check if the lower endpoint is
less
1380          A(L)=A(K)                 !  than the midpoint.  If not, the
n switch
1390          A(K)=T1                   !  the upper and lower endpoints.
1400          GOTO Middlehigh
1410 Passed: IF L-I<=J-K THEN Storehigh   ! Sort the shortest segment
first.
1420          L(M)=I                        ! Store the lower
1430          U(M)=L                        !  endpoints.
1440          I=K                           ! Set the new lower endpoint
.
1450          M=M+1                         ! Push the stack
1460          GOTO 1510
1470 Storehigh: L(M)=K                      ! Store the upper
1480            U(M)=J                      !  endpoints
1490            J=L                         ! Set the new upper endpoint
.
1500            M=M+1                       ! Push the stack
1510            IF J-I>=11 THEN Start2
1520            IF I=I1 THEN Start1
1530            I=I-1
1540 Inc: I=I+1                         ! Increment lower endpoint.
1550      IF I=J THEN Nextgroup         ! If the current segment is sorted
, then
1560      T=A(I+1)                      !  sort the next segment.
1570     IF Incdec=0 THEN B6
1580 I6:  IF A(I)<=T THEN Inc
1590     GOTO 1610
1600 B6:  IF A(I)>=T THEN Inc           ! Check to see if next element is
in order.
1610      K=I                           ! Insert element in otherwise sort
ed list.
1620 Copy: A(K+1)=A(K)                  ! This section bumps the array up.
1630       K=K-1                        ! Prepare to bump next element.
```

```
1650         IF Incdec=0 THEN b7
1660 I7:     IF T<A(K) THEN Copy
1670         GOTO 1690
1680 b7:     IF T>A(K) THEN Copy      ! Check to see if insertion is her
e.
1690         A(K+1)=T                 ! If so, then insert.
1700         GOTO Inc
1710 Nextgroup: M=M-1                 ! Pop the stack.
1720            IF M=0 THEN Out       ! Check for end conditions.
1730            I=L(M)                ! Restore the
1740            J=U(M)                !  previous endpoints.
1750            GOTO 1510
1760 Out: SUBEXIT
```

# Annotated Listing of Subprogram Vectorsort_q

```
890 SUB Vectorsort_q(A(*),I1,J1,Incdec)
900       N=J1+1-I1
910       Logtwo=INT(LGT(N)/LGT(2))+1
920       CALL Qsort(A(*),Logtwo,I1,J1,Incdec)
930 SUBEXIT
940 SUB Qsort(A(*),Log,I1,J1,Incdec)
950  OPTION BASE 1
960  DIM L(Log),U(Log)
970       M=1                                  ! Set stack pointer.
980       I=I1                                 ! Set lower endpoint.
990       J=J1                                 ! Set upper endpoint.
1000 Start1:IF I>=J THEN Nextgroup
1010 Start2:K=I
1020        I2=INT((J+I)/2)                    ! Determine the midpoint of a
 segment.
1030        T=A(I2)
1040        IF Incdec=0 THEN D1
1050 I1:    IF A(I)<=T THEN Lowmiddle1
1060        GOTO 1080.
1070 D1:    IF A(I)>=T THEN Lowmiddle1         ! Check to see if lower endpo
int and
1080        A(I2)=A(I)                         !  midpoint are in order.  If
 not,
1090        A(I)=T                             !  switch them.
1100        T=A(I2)                            ! Reset midpoint.
1110 Lowmiddle1: L=J                              ! Set upper endpoint.
1120             IF Incdec=0 THEN D2
1130 I2:         IF A(J)>=T THEN Middlehigh
1140             GOTO 1160
1150 D2:         IF A(J)<=T THEN Middlehigh     ! Check to see if the mid
point and
1160             A(I2)=A(J)                        !  the upper endpoint are
 in order.
1170             A(J)=T                            ! If not, switch them.
1180             T=A(I2)
1190             IF Incdec=0 THEN D3
1200 I3:         IF A(I)<=T THEN Middlehigh
1210             GOTO 1230
1220 D3:         IF A(I)>=T THEN Middlehigh     ! Check to see if the swi
tch left
1230             A(I2)=A(I)                        !  the lower endpoint and
 the mid-
1240             A(I)=T                            !  point in order.
1250             T=A(I2)                           ! If not, switch them.
1260 Middlehigh: L=L-1                             ! Decrement the upper end
point.
1270             IF Incdec=0 THEN D4
```

```
1280 I4:          IF A(L)>T THEN Middlehigh
1290              GOTO 1310
1300 D4:          IF A(L)<T THEN Middlehigh   ! Check to see if the new
 upper
1310              T1=A(L)                     !  endpoint is in order.
1320 Stepup: K=K+1                    ! If not, save the upper endpoint
and
1330         IF Incdec=0 THEN D5
1340 I5:     IF A(K)<T THEN Stepup    !  increment the lower endpoint.
Now
1350         GOTO 1370
1360 D5:     IF A(K)>T THEN Stepup
1370         IF K>L THEN Passed       !  check if the lower endpoint is
less
1380         A(L)=A(K)                !  than the midpoint.  If not, the
n switch
1390         A(K)=T1                  !  the upper and lower endpoints.
1400         GOTO Middlehigh
1410 Passed: IF L-I<=J-K THEN Storehigh   ! Sort the shortest segment
first.
1420         L(M)=I                        ! Store the lower
1430         U(M)=L                        !  endpoints.
1440         I=K                           ! Set the new lower endpoint
.
1450         M=M+1                         ! Push the stack
1460         GOTO 1510
1470 Storehigh: L(M)=K                     ! Store the upper
1480            U(M)=J                     !  endpoints
1490            J=L                        ! Set the new upper endpoint
.
1500            M=M+1                      ! Push the stack
1510            IF J-I>=11 THEN Start2
1520            IF I=I1 THEN Start1
1530            I=I-1
1540 Inc: I=I+1                       ! Increment lower endpoint.
1550      IF I=J THEN Nextgroup       ! If the current segment is sorted
, then
1560      T=A(I+1)                    !  sort the next segment.
1570      IF Incdec=0 THEN D6
1580 I6:  IF A(I)<=T THEN Inc
1590      GOTO 1610
1600 D6:  IF A(I)>=T THEN Inc         ! Check to see if next element is
in order.
1610      K=I                         ! Insert element in otherwise sort
ed list.
1620 Copy: A(K+1)=A(K)                ! This section bumps the array up.
1630       K=K-1                      ! Prepare to bump next element.
```

```
1650        IF Incdec=0 THEN D7
1660 I7:    IF T<A(K) THEN Copy
1670        GOTO 1690
1680 D7:    IF T>A(K) THEN Copy       ! Check to see if insertion is her
e.
1690        A(K+1)=T                  ! If so, then insert.
1700        GOTO Inc
1710 Nextgroup: M=M-1                 ! Pop the stack.
1720            IF M=0 THEN Out       ! Check for end conditions.
1730            I=L(M)                ! Restore the
1740            J=U(M)                !  previous endpoints.
1750            GOTO 1510
1760 Out: SUBEXIT
```

METHOD:

This subroutine uses a method of sorting which is similar to the QUICKERSORT algorithm by R.S. Scowen [2], which in turn is similar to an algorithm by Hibbard [3,4] and to Hoare's QUICKSORT [5]. The subroutine was translated from a FORTRAN listing of Algorithm 347 in the Communications of the ACM [1].

REFERENCES:


1. Singleton, Richard C. Certification of Algorithm 347. Comm. ACM Vol. 12, Number 3, March 1969, pp. 185-187.

2. Scowen, R.S. Algorithm 271, Quickersort. Comm. ACM 8 (Nov 1965), p. 669.

3. Hibbard, Thomas N. Some combinatorial properties of certain trees with applications to searching and sorting. Journal of ACM 9 (Jan 1962), 13-28.

4. Hibbard, Thomas N. An empirical study of minimal storage sorting. Comm. ACM 6 (May 1963), 206-213.

5. Hoare, C.A.R. Algorithms 63, Partition, and 64, Quicksort. Comm. ACM 4 (July 1961), 321.

Driver Utilization:

File name: QSORTD

This driver allows the user to either enter an array of his own, or have a random array generated. The array will then be printed, and the driver will call subroutine Vectorsort_q which will sort the array. Upon returning to the driver, the sorted array will be printed and titled.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "QSORTD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: the select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c) Go to step 5.

5. When "Enter the lower subscript of the array" appears in the display area:
   a. Enter: The lower subscript of the array.
   b. Press: CONT
6. When "Enter the upper subscript of the array" appears in the display area:
   a. Enter: The upper subscript of the array.
   b. Press: CONT
7. When "Do you want to select your own array (Y/N)?" appears in the display area:
   a. If you want to enter your own array:
      1) Enter: Y
      2) Press: CONT
      3) Go to step 8.
   b. If you do not want to enter your own array, but would rather have a random array generated:
      1) Enter: N
      2) Press: CONT
      3) Go to step 10.
8. When "ELEMENT # i?" appears in the display area:
   a. Enter: The ith element of the array.
   b. Press: CONT
   c. Repeat step 8 as often as needed.
9. When "Enter the subscript?" appears in the display area:
   a. If you wish to make any changes in the array:
      1) Enter: The subscript of the element you wish to change.
      2) Press: CONT
      3) Go to step 8.
   b. If you do not wish to make any changes in the array:
      1) Press: CONT
      2) Go to step 10.
10. When "Do you want to sort in ascending or descending order (A/D)?" appears in the display:
   a. If you want to sort in ascending order:
      1) Enter: A
      2) Press: CONT

b. If you want to sort in descending order:

1) Enter: D

2) Press: CONT

11. The sorted array will be printed and titled.

12. When "Do you want to sort the array again (Y/N)?" appears in the display area:

a. If you want to sort the array again:

1) Type: Y

2) Press: CONT

3) Go to step 9.

b. If you do not want to sort the array again:

1) Type: N

2) Press: CONT

3) The program stops.

## EXAMPLE

The following array is sorted in descending order:

```
UNSORTED ARRAY
#   -10      282
#    -9      321
#    -8      691
#    -7      422
#    -6      501
#    -5      132
#    -4      486
#    -3      169
#    -2      907
#    -1      971
#     0      191
#     1      186
#     2      200
#     3      279
#     4      315
#     5      50

SORTED ARRAY
#   -10      971
#    -9      907
#    -8      691
#    -7      501
#    -6      486
#    -5      422
#    -4      321
#    -3      315
#    -2      282
#    -1      279
#     0      200
#     1      191
#     2      186
#     3      169
#     4      132
#     5      50
```

## Driver Listing

```
10 LINK "QSORTN",1260
20 PRINTER IS 16
30 PRINT PAGE,"Any instructions or prompts that occur while this progr
am is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONT without entering"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90 S=16
100 INPUT "Printer select code?",S
110  S=INT(S)
120  IF (S>16) OR (S<0) THEN 90
130  IF S<>16 THEN 151
140  Top$="N"
150  GOTO 240
151  PRINTER IS S
152  PRINT
153  PRINTER IS 16
160  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
170  PRINT "has top-of-form generation and whether or not you wish to
make"
180  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
190  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
200  PRINT "generation, enter an N and press CONT."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 210
240  PRINT PAGE
250  PRINT "Below you are asked to enter the lower and upper subscript
s of the"
260  PRINT "array you wish to sort.  The subscripts may be either nega
tive,"
270 PRINT "positive, or both.  The only restriction is that the lower
subscript"
280  PRINT "must be less than or equal to the upper subscript",LIN(4)
290 INPUT "Enter the lower subscript of the array",L
300 INPUT "Enter the upper subscript of the array",U
310  IF U>=L THEN 360
```

```
320  BEEP
330  DISP "ILLEGAL RANGE"
340  WAIT 700
350  GOTO 290
360  PRINT PAGE
370 CALL Sortdr(L,U,S,Top$)
371  BEEP
372  DISP "PROGRAM COMPLETED"
380 END
390 SUB Sortdr(L,U,S,Top$)
400 DIM A(L:U),B(L:U)
410  RANDOMIZE
420  PRINT "If you wish to enter your own array, enter Y in response t
o the"
430  PRINT "question below.  If you don't wish to enter your own array
, but"
440  PRINT "would rather have the machine generate a random array, ent
er N."
450  PRINT "If you press CONT without entering a number, Y will be ass
umed"
460  PRINT "as the response.",LIN(4)
470  A$="Y"
480     INPUT "Do you want to select your own array (Y/N)?",A$
490     IF (A$="Y") OR (A$="y") THEN Manual
500     IF (A$="N") OR (A$="n") THEN Auto
510     GOTO 470
520 Manual: PRINT PAGE
530         PRINTER IS S
540         FOR I=L TO U
550            DISP "ELEMENT #";I;
560            INPUT A(I)
570            PRINT USING 580;I,A(I)
580            IMAGE "#",M4D,5X,K
590         NEXT I
600 PRINTER IS 16
610 IF S<>16 THEN PRINT PAGE;
620 PRINT LIN(1),"If you want to make any changes, enter the subscript
 of the"
630 PRINT "element you wish to change.  Otherwise, press CONT without"
640  PRINT "entering a number.",LIN(2)
650  PRINTER IS S
660 Input: I=3.141592654
670        INPUT "Enter the subscript",I
680        IF I=3.141592654 THEN Run
690        IF (I>=L) AND (I<=U) THEN 740
700        BEEP
710        DISP "ILLEGAL SUBSCRIPT"
```

```
720         WAIT 700
730         GOTO Input
740         DISP "ELEMENT #";I;
750         INPUT A(I)
760         PRINT USING 580;I,A(I)
770         GOTO Input
780 Auto: FOR I=L TO U
790        A(I)=INT(RND*1000)
800        NEXT I
810 Run: PRINTER IS S
820       IF (Top$="y") OR (Top$="Y") THEN PRINT PAGE;
830       PRINT LIN(2),"UNSORTED ARRAY"
840       CALL Output(A(*),L,U)
850       PRINTER IS 16
860       IF S<>16 THEN PRINT PAGE;
870       PRINT LIN(2),"If you want the array to be sorted in ascending
 order,"
880       PRINT "enter A.  If you want the array to be sorted in descen
ding"
890       PRINT "order, enter D.  If you press CONT without entering"
900       PRINT "anything, A will be the assumed response.",LIN(2)
910       A$="A"
920       INPUT "Do you want to sort in ascending or descending order (
A/D)?",A$
930       IF (UPC$(A$)="A") OR (UPC$(A$)="D") THEN 980
940       BEEP
950       DISP "ILLEGAL RESPONSE"
960       WAIT 700
970       GOTO 920
980       IF S<>16 THEN PRINT PAGE
990       PRINTER IS S
1000      Incdec=1
1010      IF (A$="D") OR (A$="d") THEN Incdec=0
1020      MAT B=A
1030      DISP "SORTING"
1040      CALL Vectorsort_q(A(*),L,U,Incdec)
1050      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE;
1060      PRINT LIN(2),"SORTED ARRAY"
1070      CALL Output(A(*),L,U)
1080      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE
1090      INPUT "Do you want to sort the array again (Y/N)?",A$
1100      IF UPC$(A$)="N" THEN 1170
1110      IF UPC$(A$)="Y" THEN 1130
1120      GOTO 1090
1130      MAT A=B
1140      PRINT LIN(2),"ORIGINAL VECTOR"
1150      CALL Output(A(*),L,U)
```

```
1160      GOTO 600
1170      PRINTER IS 16
1180 SUBEXIT
1190 SUB Output(A(*),L,U)
1200    FOR I=L TO U
1210        PRINT USING 1220;I,A(I)
1220        IMAGE "#",M4D,5X,K
1230    NEXT I
1240 SUBEXIT
1250 SUBEND
```

HEWLETT hp PACKARD

# MINIMAL STORAGE SORT OF A NUMERIC VECTOR (SHELL SORT)

Subprogram Name: Vectorsort_s

This subprogram allows the user to sort a one-dimensional array (or any subset of the array) into ascending or descending order.

This sort is slower than the Fast Sort on page 9, but it takes much less memory. The subroutine itself takes 2782 bytes less memory, and no auxiliary storage is needed, as in the Fast Sort.

Subprogram Utilization:

File name: SSORTN

Calling Syntax: CALL Vectorsort_s(A(*), I1,J1, Incdec)

Input Parameters:

A(*) - The first parameter in the parameter list must be a one-dimensional full-precision numeric vector. This is the vector which will be sorted.

I1 - The second parameter must be a full-precision variable, constant, or expression. I1 represents the lower bound of the sort.

J1 - The third parameter must be a full-precision variable, constant, or expression. J1 represents the upper bound of the sort.

Incdec- The fourth parameter must be a full-precision variable, constant, or expression. It indicates whether the sort will be ascending or descending. If Incdec = 0, the sort will be descending. If Incdec = 1, the sort will be ascending. If Incdec = anything else, the vector will not be sorted.

Output Parameters:

A(*) - After this subprogram is executed, this vector will be sorted from elements I1 through J1 (if the value passed in as Incdec is Ø or 1 (see the section under Input Parameters)).

Local Variables:

H - $2^{(S-1)}$

I - temporary subscript

J - loop counter

K - temporary storage variable (for switching)

S - counter on loop to sort every $2^{(S-1)}$ element

T - $LOG_2$ of the number of elements to be sorted

Special Considerations and Programming Hints:

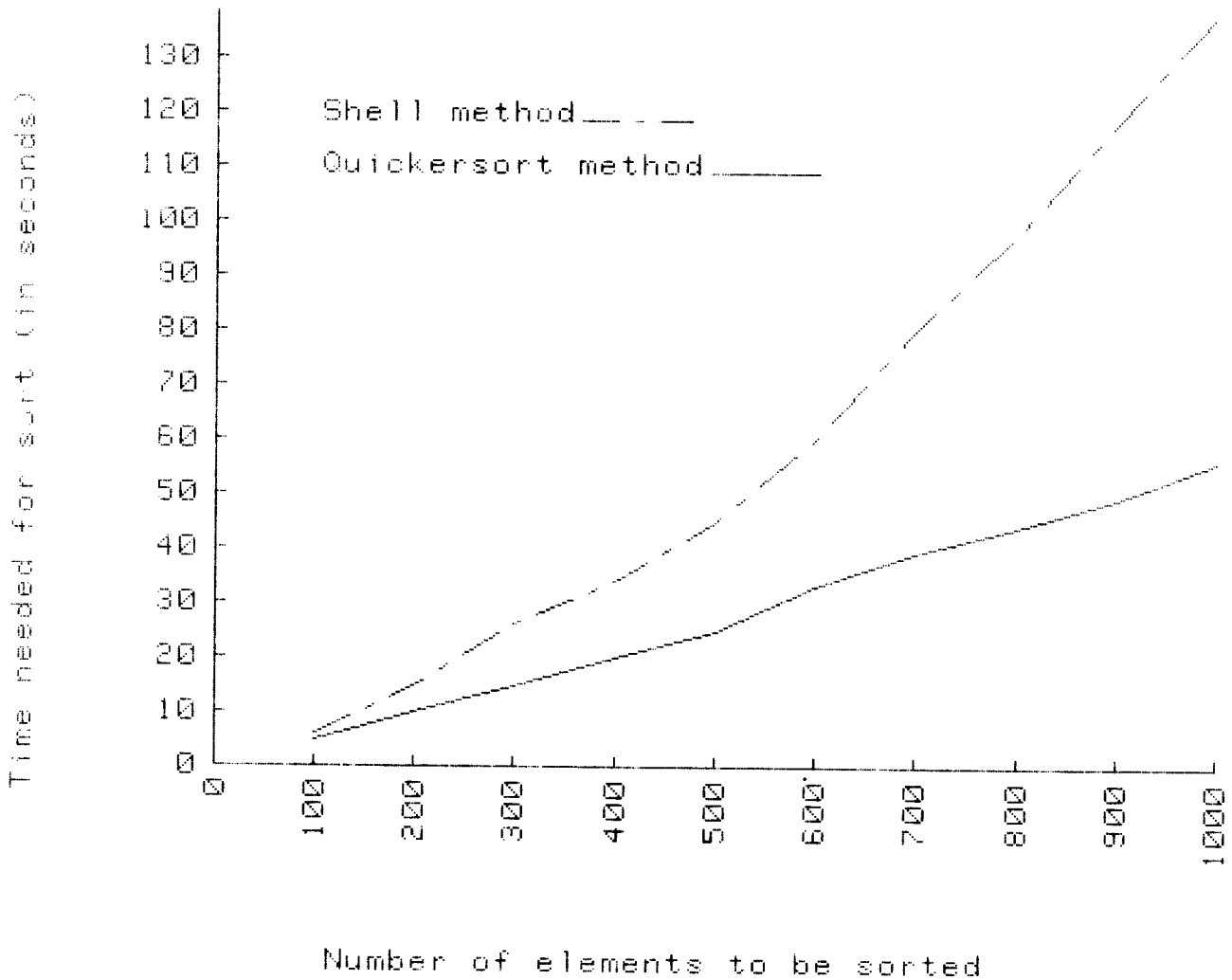
1. If the value passed in for the lower bound of the sort is greater than or equal to the value passed in for the upper bound of the sort, the array will not be changed from its input state.
2. As mentioned in the introduction, this sort takes less memory than the Fast Sort on page 9 , but it is significantly slower. If memory is not a concern, the Fast Sort is recommended. (Note: See timing comparison graph immediately following this section - GRAPH 1.)
3. This sort may be made faster by removing the capability of sorting both in ascending and descending order. An extra listing is included showing which lines to delete in order to accomplish this. The modified listing (the listing may be found immediately following this section) will sort in ascending order. To sort in descending order, delete the statement labeled "Increase:" instead of the one labeled "Decrease:" as shown in the modified listing.

   GRAPH 2 immediately following this section shows how much improvement in speed is gained by using the procedure outlined above.

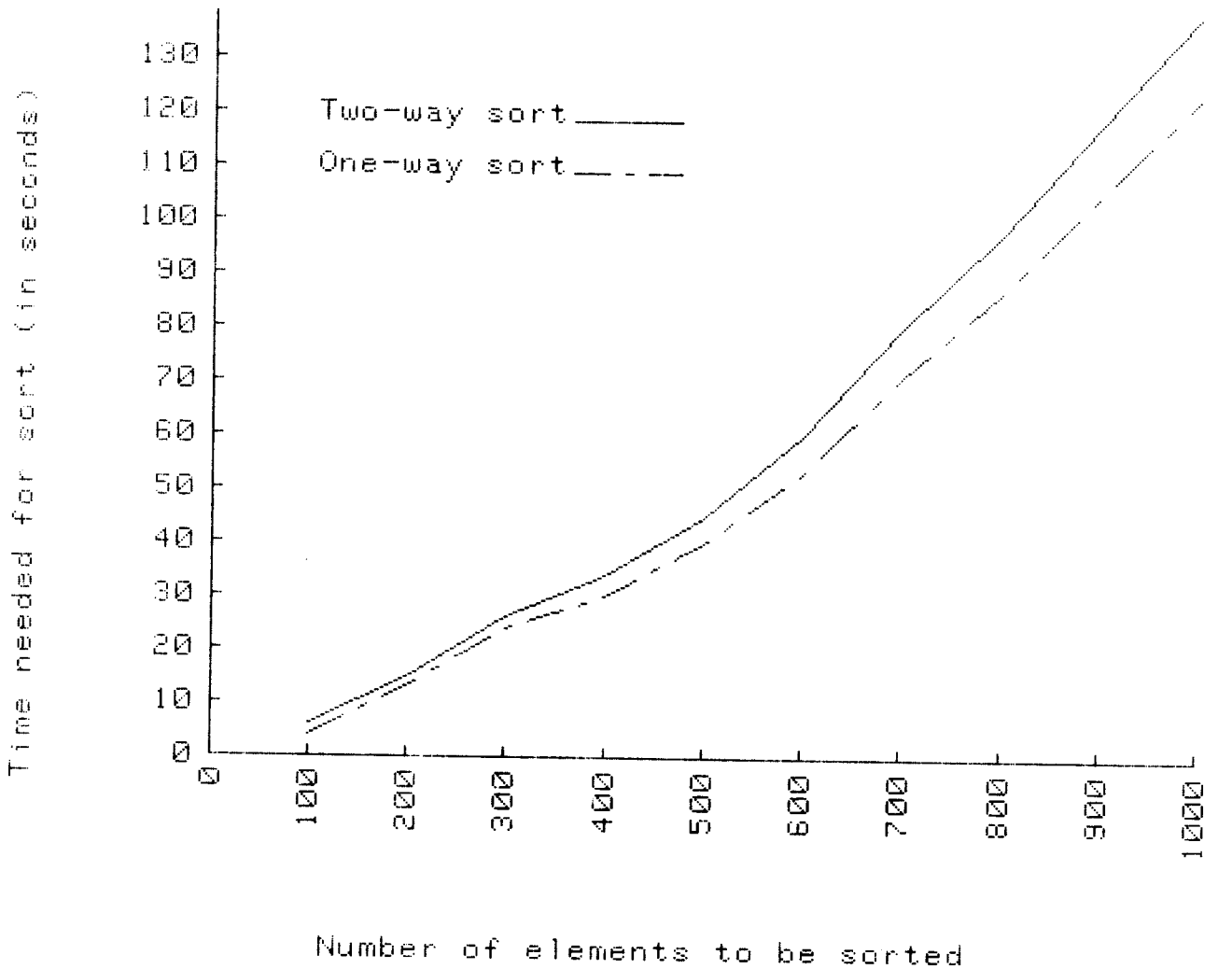
4. System Configuration:

Standard memory option
(optional) Printer

GRAPH 1 - Quickersort vs. Shell sort



GRAPH 2 - Two-way sort vs. One-way sort



## Modified Subprogram Listing (Ascending sort only)

This listing shows those lines which can be removed to make the sort faster. (The deleted lines would allow the user to sort both in ascending and descending order if they were left in.)

```
890 SUB Vectorsort_s(A(*),I1,J1,Incdec)
900  REM  This subroutine sorts a one-dimensional vector (or any subse
t of that
910  REM  vector) into either ascending or descending order.  Shell's
sort
920  REM  (diminishing increment sort) is used.
930  IF (Incdec<>0) AND (Incdec<>1) THEN SUBEXIT
940  T=INT(LOG(J1+1-I1)/LOG(2))  !  Compute the number of diminishing
increments
950                              !    to be used in the sort.
960  FOR S=T TO 1 STEP -1
970     H=2^(S-1)                !  The set of increments to be used i
s ...,16,8
980                              !    4,2,1
990     FOR J=H+I1 TO J1
1000       I=J-H
1010       K=A(J)
1020 ~~Decide:IF Incdec=0 THEN Decrease~~
1030 Increase:IF K>=A(I) THEN Insert
1040          ~~GOTO Switch~~
1050 ~~Decrease:IF K<=A(I) THEN Insert~~
1060 Switch:  A(I+H)=A(I)
1070          I=I-H
1080          IF I>=I1 THEN Decide
1090 Insert:  A(I+H)=K
1100    NEXT J
1110 NEXT S
1120 SUBEXIT
```

# Annotated Listing of Subprogram Vectorsort_s

```
890 SUB Vectorsort_s(A(*),I1,J1,Incdec)
900   REM  This subroutine sorts a one-dimensional vector (or any subse
t of that
910   REM  vector) into either ascending or descending order.  Shell s
ort
920   REM  (diminishing increment sort) is used.
930   IF (Incdec<>0) AND (Incdec<>1) THEN SUBEXIT
940   T=INT(LOG(J1+1-I1)/LOG(2))  !  Compute the number of diminishing
increments
950                               !     to be used in the sort.
960   FOR S=T TO 1 STEP -1
970      H=2^(S-1)                !  The set of increments to be used i
s ...,16,8
980                               !     4,2,1
990      FOR J=H+I1 TO J1
1000        I=J-H
1010        K=A(J)
1020 Decide:IF Incdec=0 THEN Decrease
1030 Increase:IF K>=A(I) THEN Insert
1040          GOTO Switch
1050 Decrease:IF K<=A(I) THEN Insert
1060 Switch:  A(I+H)=A(I)
1070          I=I-H
1080          IF I>=I1 THEN Decide
1090 Insert:  A(I+H)=K
1100     NEXT J
1110  NEXT S
1120  SUBEXIT
```

Method:

The method used in this subprogram is Donald Shell's dimishing increment sort. First, a sequence of diminishing increments $h_t$, $h_{t-1}$,..., $h_1$ is selected. (In this case, t is $\log_2 N$, where N is the number of elements to be sorted, and the h's are $2^{t-1}$, $2^{t-2}$, ....$2^0$.) Then, for each h, a straight insertion sort is performed on those elements of the array which are h elements apart. By the time the last h has been used, the array is sorted. (Note: the other h's may be anything, as long as they are diminishing, but $h_1$ must always be 1.)

REFERENCE:

Knuth, Donald E., THE ART OF COMPUTER PROGRAMMING, VOL. 3 (SORTING AND SEARCHING) (Addison-Wesley 1973) pp. 84-85.

Driver Utilization:

File name: SSORTD

This driver allows the user to either enter an array of his own or have a random array generated. The array will then be printed, and the driver will call the subroutine Vectorsort_s which will sort the array. Upon returning to the driver, the sorted array will be printed and titled.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SSORTD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.

         or
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c) Go to step 5.

5. When "Enter the lower subscript of the array?" appears in the display area:
   a. Enter: The lower subscript of the array.
   b. Press: CONT
6. When "Enter the upper subscript of the array?" appears in the display area:
   a. Enter: The upper subscript of the array.
   b. Press: CONT
7. When "Do you want to select your own array (Y/N)?" appears in the display area:
   a. If you want to enter your own array:
      1) Enter: Y
      2) Press: CONT
      3) Go to step 8.
   b. If you do not want to enter your own array, but would rather have a random array generated:
      1) Enter: N
      2) Press: CONT
      3) Go to step 10.
8. When "ELEMENT # i?" appears in the display area:
   a. Enter: The ith element of the array.
   b. Press: CONT
   c. Repeat step 8 as often as needed.
9. When "Enter the subscript?" appears in the display area:
   a. If you wish to make any changes in the array:
      1) Enter: The subscript of the element you wish to change.
      2) Press: CONT
      3) Go to step 8.
   b. If you do not want to make any changes in the array:
      1) Press: CONT
      2) Go to step 10.
10. When "Do you want to sort in ascending or descending order (A/D)?" appears in the display:
   a. If you want to sort in ascending order:
      1) Enter: A
      2) Press: CONT

b. If you want to sort in descending order:

1) Type: D

2) Press: CONT

11. The sorted array will be printed and titled.

12. When "Do you want to sort the array again (Y/N)?" appears in the display area:

a. If you want to sort the array again:

1) Type: Y

2) Press: CONT

3) Go to step 9.

b. If you do not want to sort the array again:

1) Type: N

2) Press: CONT

3) The program stops.

The following array is sorted in ascending order:

```
UNSORTED ARRAY
#     1      302
#     2      141
#     3      311
#     4      842
#     5      721
#     6      153
#     7      960
#     8      60
#     9      240
#    10      291
#    11      535
#    12      540
#    13      460
#    14      646
#    15      102
#    16      71
#    17      212
#    18      932
#    19      319
#    20      683


SORTED ARRAY
#     1      60
#     2      71
#     3      102
#     4      141
#     5      153
#     6      212
#     7      240
#     8      291
#     9      302
#    10      311
#    11      319
#    12      460
#    13      535
#    14      540
#    15      646
#    16      683
#    17      721
#    18      842
#    19      932
#    20      960
```

# Driver Listing

```
10 LINK "SSORTN",1260
20 PRINTER IS 16
30 PRINT PAGE,"Any instructions or prompts that occur while this progr
am is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONT without entering"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90 S=16
100 INPUT "Printer select code?",S
110  S=INT(S)
120  IF (S>16) OR (S<0) THEN 90
130  IF S<>16 THEN 151
140  Top$="N"
150  GOTO 240
151  PRINTER IS S
152  PRINT
153  PRINTER IS 16
160  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
170  PRINT "has top-of-form generation and whether or not you wish to
make"
180  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
190  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
200  PRINT "generation, enter an N and press CONT."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 210
240  PRINT PAGE
250  PRINT "Below you are asked to enter the lower and upper subscript
s of the"
260  PRINT "array you wish to sort.  The subscripts may be either nega
tive,"
270 PRINT "positive, or both.  The only restriction is that the lower
subscript"
280  PRINT "must be less than or equal to the upper subscript",LIN(4)
290 INPUT "Enter the lower subscript of the array",L
300 INPUT "Enter the upper subscript of the array",U
310  IF U>=L THEN 360
```

```
320  BEEP
330  DISP "ILLEGAL RANGE"
340  WAIT 700
350  GOTO 290
360  PRINT PAGE
370 CALL Sortdr(L,U,S,Top$)
371 BEEP
372  DISP "PROGRAM COMPLETED"
380 END
390 SUB Sortdr(L,U,S,Top$)
400 DIM A(L:U),B(L:U)
410  RANDOMIZE
420  PRINT "If you wish to enter your own array, enter Y in response t
o the"
430  PRINT "question below.  If you don't wish to enter your own array
, but"
440  PRINT "would rather have the machine generate a random array, ent
er N."
450  PRINT "If you press CONT without entering a number, Y will be ass
umed"
460  PRINT "as the response.",LIN(4)
470  A$="Y"
480     INPUT "Do you want to select your own array (Y/N)?",A$
490     IF (A$="Y") OR (A$="y") THEN Manual
500     IF (A$="N") OR (A$="n") THEN Auto
510     GOTO 470
520 Manual: PRINT PAGE
530         PRINTER IS S
540         FOR I=L TO U
550            DISP "ELEMENT #";I;
560            INPUT A(I)
570            PRINT USING 580;I,A(I)
580            IMAGE "#",M4D,5X,K
590         NEXT I
600 PRINTER IS 16
610 IF S<>16 THEN PRINT PAGE;
620 PRINT LIN(1),"If you want to make any changes, enter the subscript
 of the"
630 PRINT "element you wish to change.  Otherwise, press CONT without"
640  PRINT "entering a number.",LIN(2)
650  PRINTER IS S
660 Input: I=3.141592654
670        INPUT "Enter the subscript",I
680        IF I=3.141592654 THEN Run
690        IF (I>=L) AND (I<=U) THEN 740
700        BEEP
710        DISP "ILLEGAL SUBSCRIPT"
```

```
720         WAIT 700
730         GOTO Input
740         DISP "ELEMENT #";I;
750         INPUT A(I)
760         PRINT USING 580;I,A(I)
770         GOTO Input
780 Auto:  FOR I=L TO U
790        A(I)=INT(RND*1000)
800        NEXT I
810 Run:  PRINTER IS S
820       IF (Top$="y") OR (Top$="Y") THEN PRINT PAGE;
830       PRINT LIN(2),"UNSORTED ARRAY"
840       CALL Output(A(*),L,U)
850       PRINTER IS 16
860       IF S<>16 THEN PRINT PAGE;
870       PRINT LIN(2),"If you want the array to be sorted in ascending
 order,"
880       PRINT "enter A.  If you want the array to be sorted in descen
ding"
890       PRINT "order, enter D.  If you press CONT without entering"
900       PRINT "anything, A will be the assumed response.",LIN(2)
910       A$="A"
920       INPUT "Do you want to sort in ascending or descending order (
A/D)?",A$
930       IF (UPC$(A$)="A") OR (UPC$(A$)="D") THEN 980
940       BEEP
950       DISP "ILLEGAL RESPONSE"
960       WAIT 700
970       GOTO 920
980       IF S<>16 THEN PRINT PAGE
990       PRINTER IS S
1000      Incdec=1
1010      IF (A$="D") OR (A$="d") THEN Incdec=0
1020      MAT B=A
1030      DISP "SORTING"
1040      CALL Vectorsort_s(A(*),L,U,Incdec)
1050      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE;
1060      PRINT LIN(2),"SORTED ARRAY"
1070      CALL Output(A(*),L,U)
1080      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE
1090      INPUT "Do you want to sort the array again (Y/N)?",A$
1100      IF UPC$(A$)="N" THEN 1170
1110      IF UPC$(A$)="Y" THEN 1130
1120      GOTO 1090
1130      MAT A=B
1140      PRINT LIN(2),"ORIGINAL VECTOR"
1150      CALL Output(A(*),L,U)
```

```
1160      GOTO 600
1170      PRINTER IS 16
1180 SUBEXIT
1190 SUB Output(A(*),L,U)
1200   FOR I=L TO U
1210        PRINT USING 1220;I,A(I)
1220        IMAGE "#",M4D,5X,K
1230   NEXT I
1240 SUBEXIT
1250 SUBEND
```

Subprogram Name: Arraysort_q

This subprogram allows the user to sort a two-dimensional array on any row or any column into ascending or descending order, maintaining column or row integrity, as the case may be.

This sort is faster than the Minimal Storage Sort of a Numeric Array, but it takes up 4908 bytes more memory for the subprogram itself, in addition to the memory needed for two auxiliary storage arrays.

Subprogram Utilization:

File name: QSORTA

Calling Syntax: CALL Arraysort_q(A(*), A,B,C,D,E,F,G)

Input Parameters:

A(*) - The first parameter in the parameter list must be a two-dimensional full-precision numeric array. This is the array which will be sorted.

A - The second parameter must be a full-precision variable, constant, or expression. A is the lower row bound of the sort.

B - The third parameter must be a full-precision variable, constant, or expression. B is the upper row bound of the sort.

C - The fourth parameter must be a full-precision variable, constant, or expression. C is the lower column bound of the sort.

D - The fifth parameter must be a full-precision variable, constant, or expression. D is the upper column bound of the sort.

E - The sixth parameter must be a full-precision variable, constant, or expression. E tells whether the array A(*) is to be sorted by row or column. If E is 1, then the array will be sorted by row. If E is Ø, then the array will be sorted by column. If E is anything else, no sorting will be done.

F - The seventh parameter must be a full-precision variable, constant, or expression. F tells which row or column is to be sorted.

G - The eighth parameter must be a full-precision variable, constant, or expression. G tells whether the array A(*) is to be sorted in increasing or decreasing order. If G is 1, then the array will be sorted in increasing order. If G is Ø, then A(*) will be sorted in decreasing order. If G is anything else, then no sorting will be done.

Output Parameters:

A(*) - After this subprogram is executed, A(*) (a full-precision two-dimensional array) will be sorted between columns C and D and between rows A and B (see the section under Input Parameters for explanations of A, B, C, and D).

Local Variables:

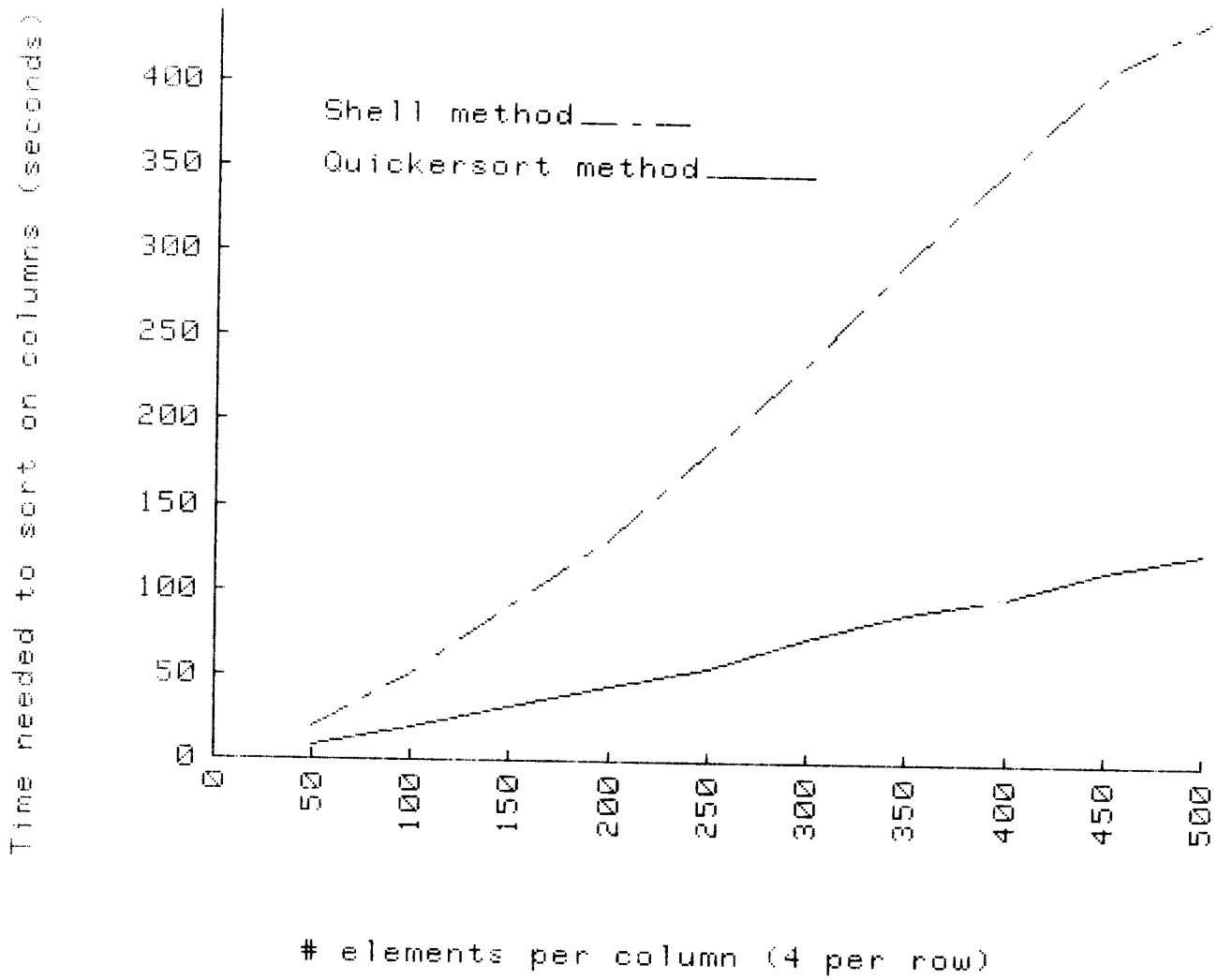
Count - loop counter
I - lower endpoint of array segment
I2 - midpoint of array segment
J - upper endpoint of array segment
K - temporary storage for I
L - temporary storage for J
L(*) - stack for lower endpoints of array segments
M - stack pointer
S(*) - temporary row (or column) storage
T - temporary storage for array element (for switching)
T(*) - temporary row (or column) storage
U(*) - stack for upper endpoints of array segments

Special Considerations and Programming Hints:

1. If the lower row bound of the sort is greater than the upper row bound, or if the lower column bound is greater than the upper column bound, the array will be unchanged from its input state.
2. As mentioned in the introduction, this sort takes more memory than the Minimal Storage Sort (Shell sort) on page 69 but it is also much faster. However, if a memory overflow occurs while running this subprogram, it may be necessary to use the Minimal Storage Sort instead. (Note: See the timing comparison graph immediately following this section—GRAPH 1.)
3. System Configuration:

   Standard Memory Option
   (Optional) Printer

GRAPH 1 - Quickersort vs. Shell sort



# Annotated Listing of Subprogram Arraysort_q

```
1280 SUB Arraysort_q(A(*),N1,N2,M1,M2,C1,W,Incdec)
1281       IF (N1>N2) OR (M1>M2) THEN SUBEXIT
1290       IF C1=1 THEN Row
1300       IF C1=0 THEN Column
1310       SUBEXIT
1320 Row: CALL Orcsort(A(*),M1,M2,N1,N2,C1,W,Incdec)
1380       SUBEXIT
1390 Column:CALL Orcsort(A(*),N1,N2,M1,M2,C1,W,Incdec)
1400         SUBEXIT
1470 SUBEXIT
1480 SUB Orcsort(A(*),Sortlow,Sorthigh,Carrylow,Carryhigh,C1,W,Incdec)
1490 OPTION BASE 1
1500 INTEGER L(LOG(Sorthigh+1-Sortlow)/LOG(2)+1),U(LOG(Sorthigh+1-Sort
low)/LOG(2)+1)
1501 DIM S(Carrylow:Carryhigh),T(Carrylow:Carryhigh)
1502 IF (W<Carrylow) OR (W>Carryhigh) THEN SUBEXIT
1510      M=1                          ! Set stack pointer.
1520      I=Sortlow                    ! Set lower endpoint.
1530      J=Sorthigh                   ! Set upper endpoint.
1540 Start1:IF I>=J THEN Nextgroup
1550 Start2:K=I
1560         I2=INT((J+I)/2)           ! Determine the midpoint of the seg
ment
1570         T=A(C1*W+NOT C1*I2,NOT C1*W+C1*I2)
1571         IF Incdec=0 THEN D1
1580 I1:     IF A(C1*W+NOT C1*I,NOT C1*W+C1*I)<=T THEN Lowmiddle1
1581         GOTO 1620
1585 D1:     IF A(C1*W+NOT C1*I,NOT C1*W+C1*I)>=T THEN Lowmiddle1
1590 !                                  Check to see if lower endpoint an
d
1600 !                                   midpoint are in order.  If not,
1610 !                                   switch them.  If so, the branch.
1620         FOR Count=Carrylow TO Carryhigh
1640               T=A(Count*C1+NOT C1*I2,NOT C1*Count+C1*I2)
1650               A(C1*Count+NOT C1*I2,NOT C1*Count+C1*I2)=A(C1*Count+
NOT C1*I,NOT C1*Count+C1*I)
1660               A(C1*Count+NOT C1*I,NOT C1*Count+C1*I)=T
1680         NEXT Count
1690         T=A(C1*W+NOT C1*I2,NOT C1*W+C1*I2) ! Reset midpoint.
1700 Lowmiddle1:L=J                   ! Set upper endpoint.
1701            IF Incdec=0 THEN D2
1710 I2:        IF A(C1*W+NOT C1*J,NOT C1*W+C1*J)>=T THEN Middlehigh
1711            GOTO 1750
1715 D2:        IF A(C1*W+NOT C1*J,NOT C1*W+C1*J)<=T THEN Middlehigh
1720 !                                  Check to see if the midpoint and
the
1730 !                                   upper endpoint are in order.  If
```

```
 not,
1740                                        switch them.  If so, then branch
.
1750            FOR Count=Carrylow TO Carryhigh
1770                  T=A(C1*Count+NOT C1*I2,NOT C1*Count+C1*I2)
1780                  A(C1*Count+NOT C1*I2,NOT C1*Count+C1*I2)=A(C1*Cou
nt+NOT C1*J,NOT C1*Count+C1*J)
1790                  A(C1*Count+NOT C1*J,NOT C1*Count+C1*J)=T
1810            NEXT Count
1820            T=A(C1*W+NOT C1*I2,NOT C1*W+C1*I2)
1821            IF Incdec=0 THEN D3
1830 I3:        IF A(C1*W+NOT C1*I,NOT C1*W+C1*I)<=T THEN Middlehigh
1831            GOTO 1880
1835 D3:        IF A(C1*W+NOT C1*I,NOT C1*W+C1*I)>=T THEN Middlehigh
1840 !                                    Check to see if the switch left t
he
1850 !                                     lower endpoint and the midpoint
in
1860 !                                     order.  If not, switch them.  If
 so,
1870 !                                     then branch.
1880            FOR Count=Carrylow TO Carryhigh
1900                  T=A(C1*Count+NOT C1*I2,NOT C1*Count+C1*I2)
1910                  A(C1*Count+NOT C1*I2,NOT C1*Count+C1*I2)=A(C1*Cou
nt+NOT C1*I,NOT C1*Count+C1*I)
1920                  A(C1*Count+NOT C1*I,NOT C1*Count+C1*I)=T
1940            NEXT Count
1950           T=A(C1*W+NOT C1*I2,NOT C1*W+C1*I2)
1960 Middlehigh: L=L-1                    ! Decrement the upper endpoint.
1961              IF Incdec=0 THEN D4
1970 I4:          IF A(C1*W+NOT C1*L,NOT C1*W+C1*L)>T THEN Middlehigh
1971              GOTO 2010
1975 D4:          IF A(C1*W+NOT C1*L,NOT C1*W+C1*L)<T THEN Middlehigh
1980 !                                    Check to see if the new upper end
point
1990 !                                     is in order.  If not, then save
the
2000 !                                     upper endpoint's row (or column)
.
2010              FOR Count=Carrylow TO Carryhigh
2030                 S(Count)=A(C1*Count+NOT C1*L,NOT C1*Count+C1*L)
2050              NEXT Count
2060 Stepup: K=K+1                        ! Increment the lower endpoint.
2061         IF Incdec=0 THEN D5
2070 I5:     IF A(C1*W+NOT C1*K,NOT C1*W+C1*K)<T THEN Stepup
2071         GOTO 2110
2075 D5:     IF A(C1*W+NOT C1*K,NOT C1*W+C1*K)>T THEN Stepup
```

```
2080 !                                  Now check if the lower endpoint i
s less
2090 !                                   than the midpoint.  If not, then
 switch
2100 !                                   the upper and lower endpoints.
2110          IF K>L THEN Passed
2120          FOR Count=Carrylow TO Carryhigh
2140               A(C1*Count+NOT C1*L,NOT C1*Count+C1*L)=A(C1*Count+N
OT C1*K,NOT C1*Count+C1*K)
2150               A(C1*Count+NOT C1*K,NOT C1*Count+C1*K)=S(Count)
2170          NEXT Count
2180          GOTO Middlehigh
2190 Passed: IF L-I<=J-K THEN Storehigh  ! Sort the shortest segment f
irst.
2200          L(M)=I                    ! Store the lower
2210          U(M)=L                    !  endpoints.
2220          I=K                       ! Set the new lower endpoint.
2230          M=M+1                     ! Push the stack.
2240          GOTO 2290
2250 Storehigh: L(M)=K                  ! Store the upper
2260            U(M)=J                  !  endpoints.
2270            J=L                     ! Set the new upper endpoint.
2280            M=M+1                   ! Push the stack.
2290            IF J-I>=11 THEN Start2
2300            IF I=Sortlow THEN Start1
2310            I=I-1
2320 Inc: I=I+1                         ! Increment lower endpoint.
2330      IF I=J THEN Nextgroup         ! If the curent segmet is sorted, t
hen
2340                                    !  sort the next segment.
2350      FOR Count=Carrylow TO Carryhigh
2370           T(Count)=A(C1*Count+NOT C1*(I+1),NOT C1*Count+C1*(I+1)
)
2390      NEXT Count
2400      T=A(C1*W+NOT C1*(I+1),NOT C1*W+C1*(I+1))
2401      IF Incdec=0 THEN D6
2410 I6:  IF A(C1*W+NOT C1*I,NOT C1*W+C1*I)<=T THEN Inc
2411      GOTO 2440
2415 D6:  IF A(C1*W+NOT C1*I,NOT C1*W+C1*I)>=T THEN Inc
2420                                    ! Check to see if the next element
is in
2430                                    !  order.
2440      K=I
2450 REM  Insert element into otherwise sorted list.
2460 Copy: FOR Count=Carrylow TO Carryhigh  !This section bumps the ar
ray up.
2480            A(C1*Count+NOT C1*(K+1),NOT C1*Count+C1*(K+1))=A(C1*C
```

```
ount+NOT C1*K,NOT C1*Count+C1*K)
2500        NEXT Count
2510        K=K-1                       ! Prepare to bump next element.
2521        IF Incdec=0 THEN D7
2530 I7:    IF T<A(C1*W+NOT C1*K,NOT C1*W+C1*K) THEN Copy
2531        GOTO 2560
2535 D7:    IF T>A(C1*W+NOT C1*K,NOT C1*W+C1*K) THEN Copy
2540                                    ! Check to see if element belongs h
ere.
2550                                    ! If so, then insert.
2560        FOR Count=Carrylow TO Carryhigh
2580              A(C1*Count+NOT C1*(K+1),NOT C1*Count+C1*(K+1))=T(Coun
t)
2600        NEXT Count
2610        GOTO Inc
2620 Nextgroup: M=M-1                   ! Pop the stack.
2630            IF M=0 THEN Out         ! Check for end conditions.
2640            I=L(M)                  ! Restore the previous
2650            J=U(M)                  !  endpoints.
2660            GOTO 2290
2670 Out: SUBEXIT
```

Method:

This subroutine uses a method of sorting which is similar to the QUICKERSORT algorithm by R. S. Scowen [Q], which in turn is similar to an algorithm by Hibbard [3,4] and to Hoare's QUICKSORT [5]. The subroutine was translated from a FORTRAN listing of Algorithm 347 in the Communications of the ACM [1].

REFERENCES:


1. Singleton, Richard C. Certification of Algorithm 347. Comm. ACM Vol. 12, Number 3, March 1969, pp. 185-187.
2. Scowen, R.S. Algorithm 271, Quickersort. Comm. ACM 8 (Nov 1965), p. 669.
3. Hibbard, Thomas N. Some combinatorial properties of certain trees with applications to searching and sorting. Journal of ACM 9 (Jan 1962), 13-28.
4. Hibbard, Thomas N. An empirical study of minimal storage sorting. Comm. ACM 6 (May 1963), 206-213.
5. Hoare, C.A.R. Algorithms 63, Partition, and 64, Quicksort. Comm. ACM 4 (July 1961), 321.

Driver Utilization:

File name: QSRTAD

This driver allows the user to either enter an array of his own, or have a random array generated. The array will then be printed, and the user will be given a choice of sorting either by row or by column, and in either descending or ascending order. The driver will then call the subprogram Arraysort_q which will sort the array. Upon returning to the driver, the sorted array will be printed and titled.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "QSRTAD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c) Go to step 5.

5. When "Enter the lower row subscript" appears in the display area:
   a. Enter: The lower row boundary of the array.
   b. Press: CONT
6. When "Enter the upper row subscript" appears in the display area:
   a. Enter: The upper row boundary of the array.
   b. Press: CONT
7. When "Enter the lower column subscript" appears in the display area:
   a. Enter: The lower column boundary.
   b. Press: CONT
8. When "Enter the upper column subscript" appears in the display area:
   a. Enter: The upper column boundary of the array.
   b. Press: CONT
9. When "Enter your own matrix (Y/N)?" appears in the display area:
   a. If you wish to enter your own matrix:
      1) Enter: Y
      2) Press: CONT
      3) Go to step 10.

      or

   a. If you do not wish to enter your own matrix, but would rather have the machine generate one for you:
      1) Enter: N
      2) Press: CONT
      3) Go to step 12.
10. When "ELEMENT # i, j?" appears in the display area:
   a. Enter: The matrix element belonging in the jth column of the ith row.
   b. Press: CONT
   c. Repeat step 10 as often as necessary.

11. When "Enter the subscripts" appears in the display area:
    a. If you wish to make any changes in the matrix:
        1) Enter: The subscripts of the element you wish to change, separated by a comma.
        2) Press: CONT
        3) Go to step 10.

        or

    a. If you do not wish to make any changes in the matrix:
        1) Press: CONT
        2) Go to step 12.
12. When "Sort by row or column (R/C)?" appears in the display area:
    a. If you want to sort by row:
        1) Enter: R
        2) Press: CONT
    b. When "Which row?" appears in the display area:
        1) Enter: The row number you want to sort on.
        2) Press: CONT

        or

    a. If you want to sort by column:
        1) Enter: C
        2) Press: CONT
    b. When "Which column?" appears in the display area:
        1) Enter: The column number you want to sort on
        2) Press: CONT
13. When "Do you want to sort in ascending or descending order (A/D)?" appears in the display area:
    a. If you wish to sort in ascending order:
        1) Enter: A
        2) Press: CONT

        or

    a. If you wish to sort in descending order:
        1) Enter: D
        2) Press: CONT

14. The sorted array will be printed and titled.
15. When "Would you like to sort the array differently (Y/N)?" appears in the display area:
    a. If you would like to sort the array differently:
        1) Type: Y
        2) Press: CONT
        3) Go to step 11.

        or

    a. If you do not want to sort the array again:
        1) Type: N
        2) Press: CONT
        3) The program stops.

UNSORTED ARRAY

| | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|
| 1 | 8.52E+02 | 6.91E+02 | 8.61E+02 | 3.93E+02 | 8.06E+02 | 6.90E+01 | 4.00E+01 |
| 2 | 1.70E+01 | 7.22E+02 | 3.16E+02 | 5.04E+02 | 2.05E+02 | 9.24E+02 | 1.62E+02 |
| 3 | 6.72E+02 | 8.70E+02 | 7.10E+01 | 4.59E+02 | 1.30E+02 | 6.31E+02 | 9.51E+02 |
| 4 | 3.10E+01 | 1.19E+02 | 8.79E+02 | 7.17E+02 | 6.18E+02 | 5.74E+02 | 1.24E+02 |
| 5 | 9.05E+02 | 1.53E+02 | 7.02E+02 | 4.42E+02 | 7.60E+02 | 8.96E+02 | 9.28E+02 |
| 6 | 8.48E+02 | 7.85E+02 | 2.58E+02 | 3.00E+01 | 2.10E+01 | 4.92E+02 | 3.74E+02 |
| 7 | 8.07E+02 | 5.12E+02 | 9.33E+02 | 4.80E+02 | 8.08E+02 | 2.81E+02 | 4.70E+01 |
| 8 | 1.10E+01 | 5.80E+01 | 3.82E+02 | 3.15E+02 | 1.18E+02 | 8.91E+02 | 9.22E+02 |
| 9 | 8.97E+02 | 4.00E+02 | 4.37E+02 | 8.80E+01 | 6.52E+02 | 7.20E+01 | 6.04E+02 |
| 10 | 2.86E+02 | 7.54E+02 | 5.14E+02 | 7.00E+01 | 5.75E+02 | 8.59E+02 | 7.26E+02 |
| 11 | 6.60E+01 | 7.30E+01 | 6.45E+02 | 7.58E+02 | 5.87E+02 | 3.28E+02 | 8.60E+02 |
| 12 | 5.40E+01 | 7.10E+02 | 9.37E+02 | 3.09E+02 | 6.07E+02 | 1.50E+01 | 6.29E+02 |
| 13 | 3.79E+02 | 8.24E+02 | 7.38E+02 | 6.83E+02 | 8.14E+02 | 7.43E+02 | 5.24E+02 |
| 14 | 1.99E+02 | 4.36E+02 | 4.20E+01 | 4.51E+02 | 3.41E+02 | 4.27E+02 | 2.63E+02 |
| 15 | 4.01E+02 | 7.30E+01 | 6.60E+01 | 8.42E+02 | 7.94E+02 | 3.24E+02 | 8.16E+02 |
| 16 | 8.07E+02 | 2.31E+02 | 5.16E+02 | 2.73E+02 | 3.75E+02 | 7.29E+02 | 5.24E+02 |
| 17 | 4.45E+02 | 8.95E+02 | 3.40E+01 | 6.84E+02 | 2.57E+02 | 2.38E+02 | 8.94E+02 |
| 18 | 5.65E+02 | 7.62E+02 | 9.77E+02 | 3.54E+02 | 4.50E+02 | 2.89E+02 | 4.90E+02 |
| 19 | 2.29E+02 | 3.22E+02 | 9.94E+02 | 5.52E+02 | 7.36E+02 | 5.07E+02 | 4.55E+02 |
| 20 | 1.70E+02 | 3.56E+02 | 7.42E+02 | 6.57E+02 | 2.13E+02 | 7.14E+02 | 1.13E+02 |
| 21 | 1.21E+02 | 8.72E+02 | 1.62E+02 | 5.85E+02 | 1.53E+02 | 8.60E+02 | 8.86E+02 |
| 22 | 8.28E+02 | 4.53E+02 | 2.62E+02 | 1.70E+01 | 8.55E+02 | 8.41E+02 | 3.80E+01 |
| 23 | 3.81E+02 | 7.56E+02 | 9.18E+02 | 3.31E+02 | 7.90E+02 | 5.32E+02 | 8.53E+02 |
| 24 | 7.47E+02 | 5.75E+02 | 5.84E+02 | 3.11E+02 | 2.28E+02 | 4.70E+01 | 8.75E+02 |
| 25 | 1.21E+02 | 9.32E+02 | 1.22E+02 | 7.92E+02 | 8.31E+02 | 3.63E+02 | 9.57E+02 |
| 26 | 2.01E+02 | 1.44E+02 | 2.54E+02 | 2.26E+02 | 1.10E+02 | 8.90E+01 | 9.20E+01 |
| 27 | 6.45E+02 | 4.60E+02 | 4.22E+02 | 3.25E+02 | 7.38E+02 | 3.23E+02 | 9.62E+02 |
| 28 | 7.07E+02 | 1.09E+02 | 3.60E+02 | 4.23E+02 | 2.97E+02 | 8.77E+02 | 6.16E+02 |
| 29 | 7.24E+02 | 7.23E+02 | 7.68E+02 | 6.24E+02 | 1.03E+02 | 7.86E+02 | 7.92E+02 |
| 30 | 3.10E+02 | 8.88E+02 | 1.78E+02 | 3.46E+02 | 1.82E+02 | 1.04E+02 | 9.30E+02 |
| 31 | 1.86E+02 | 8.60E+02 | 3.10E+01 | 4.51E+02 | 5.12E+02 | 1.82E+02 | 7.80E+01 |
| 32 | 1.61E+02 | 9.77E+02 | 8.76E+02 | 2.93E+02 | 7.05E+02 | 2.83E+02 | 6.33E+02 |
| 33 | 5.46E+02 | 4.71E+02 | 2.74E+02 | 2.20E+02 | 5.86E+02 | 9.68E+02 | 5.83E+02 |
| 34 | 4.60E+01 | 9.82E+02 | 8.80E+02 | 9.14E+02 | 3.59E+02 | 2.70E+01 | 9.74E+02 |
| 35 | 1.77E+02 | 1.04E+02 | 7.46E+02 | 6.85E+02 | 4.90E+02 | 2.19E+02 | 8.20E+02 |

The array is sorted on column 9 in descending order.

SORTED ARRAY

| | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|
| 1 | 4.60E+01 | 9.82E+02 | 8.80E+02 | 9.14E+02 | 3.59E+02 | 2.70E+01 | 9.74E+02 |
| 2 | 1.61E+02 | 9.77E+02 | 8.76E+02 | 2.93E+02 | 7.05E+02 | 2.83E+02 | 6.33E+02 |
| 3 | 1.21E+02 | 9.32E+02 | 1.22E+02 | 7.92E+02 | 8.31E+02 | 3.63E+02 | 9.57E+02 |
| 4 | 4.45E+02 | 8.95E+02 | 3.40E+01 | 6.84E+02 | 2.57E+02 | 2.38E+02 | 8.80E+02 |
| 5 | 3.10E+02 | 8.88E+02 | 1.78E+02 | 3.46E+02 | 1.82E+02 | 1.04E+02 | 9.30E+02 |
| 6 | 1.21E+02 | 3.72E+02 | 1.62E+02 | 5.85E+02 | 1.53E+02 | 8.60E+02 | 8.86E+02 |
| 7 | 6.72E+02 | 8.70E+02 | 7.10E+01 | 4.59E+02 | 1.30E+02 | 6.31E+02 | 9.51E+02 |
| 8 | 1.86E+02 | 8.60E+02 | 3.10E+01 | 4.51E+02 | 5.12E+02 | 1.82E+02 | 7.80E+01 |
| 9 | 3.79E+02 | 8.24E+02 | 7.38E+02 | 6.83E+02 | 8.14E+02 | 7.43E+02 | 5.24E+02 |
| 10 | 8.48E+02 | 7.85E+02 | 2.58E+02 | 3.00E+01 | 2.10E+01 | 4.92E+02 | 3.74E+02 |
| 11 | 5.65E+02 | 7.62E+02 | 9.77E+02 | 3.54E+02 | 4.50E+02 | 2.89E+02 | 4.90E+01 |
| 12 | 3.81E+02 | 7.56E+02 | 9.18E+02 | 3.31E+02 | 7.90E+02 | 5.32E+02 | 9.53E+02 |
| 13 | 2.86E+02 | 7.54E+02 | 5.14E+02 | 7.00E+01 | 5.75E+02 | 8.59E+02 | 7.26E+02 |
| 14 | 7.24E+02 | 7.23E+02 | 7.68E+02 | 6.24E+02 | 1.03E+02 | 7.86E+02 | 7.92E+02 |
| 15 | 1.70E+01 | 7.22E+02 | 3.16E+02 | 5.04E+02 | 2.05E+02 | 9.24E+02 | 1.62E+02 |
| 16 | 5.40E+01 | 7.10E+02 | 9.37E+02 | 3.09E+02 | 6.07E+02 | 1.50E+01 | 6.29E+02 |
| 17 | 8.52E+02 | 6.91E+02 | 8.61E+02 | 3.93E+02 | 8.06E+02 | 6.90E+01 | 4.00E+01 |
| 18 | 7.47E+02 | 5.75E+02 | 5.84E+02 | 3.11E+02 | 2.28E+02 | 4.70E+01 | 8.75E+02 |
| 19 | 8.07E+02 | 5.12E+02 | 9.33E+02 | 4.80E+02 | 8.08E+02 | 2.81E+02 | 4.70E+01 |
| 20 | 5.46E+02 | 4.71E+02 | 2.74E+02 | 2.20E+02 | 5.86E+02 | 9.68E+02 | 5.83E+02 |
| 21 | 6.45E+02 | 4.60E+02 | 4.22E+02 | 3.25E+02 | 7.38E+02 | 3.23E+02 | 9.62E+02 |
| 22 | 8.28E+02 | 4.53E+02 | 2.62E+02 | 1.70E+01 | 8.55E+02 | 8.41E+02 | 3.80E+01 |
| 23 | 1.99E+02 | 4.36E+02 | 4.20E+01 | 4.51E+02 | 3.41E+02 | 4.27E+02 | 2.63E+02 |
| 24 | 8.97E+02 | 4.00E+02 | 4.37E+02 | 8.80E+01 | 6.52E+02 | 7.20E+01 | 6.04E+02 |
| 25 | 1.70E+02 | 3.56E+02 | 7.42E+02 | 6.57E+02 | 2.13E+02 | 7.14E+02 | 1.13E+02 |
| 26 | 2.29E+02 | 3.22E+02 | 9.94E+02 | 5.52E+02 | 7.36E+02 | 5.07E+02 | 4.55E+02 |
| 27 | 8.07E+02 | 2.31E+02 | 5.16E+02 | 2.73E+02 | 3.75E+02 | 7.29E+02 | 5.24E+02 |
| 28 | 9.05E+02 | 1.53E+02 | 7.02E+02 | 4.42E+02 | 7.60E+02 | 8.96E+02 | 9.28E+02 |
| 29 | 2.01E+02 | 1.44E+02 | 2.54E+02 | 2.26E+02 | 1.10E+02 | 8.90E+01 | 9.20E+01 |
| 30 | 3.10E+01 | 1.19E+02 | 8.79E+02 | 7.17E+02 | 6.18E+02 | 5.74E+02 | 1.24E+02 |
| 31 | 7.07E+02 | 1.09E+02 | 3.60E+02 | 4.23E+02 | 2.97E+02 | 8.77E+02 | 6.16E+02 |
| 32 | 1.77E+02 | 1.04E+02 | 7.46E+02 | 6.85E+02 | 4.90E+02 | 2.19E+02 | 8.20E+02 |
| 33 | 6.60E+01 | 7.30E+01 | 6.45E+02 | 7.58E+02 | 5.87E+02 | 3.28E+02 | 8.60E+02 |
| 34 | 4.01E+02 | 7.30E+01 | 6.60E+01 | 8.42E+02 | 7.94E+02 | 3.24E+02 | 8.16E+02 |
| 35 | 1.19E+01 | 5.80E+01 | 3.82E+02 | 3.15E+02 | 1.18E+02 | 8.91E+02 | 9.22E+02 |

# Driver Listing

```
10   LINK "@SORTA",2030
20   PRINTER IS 16
30   PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is"
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONT without entering'
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90   S=16
100  INPUT "Printer select code?",S
110  S=INT(S)
120  IF (S>16) OR (S<0) THEN 90
130  IF S<>16 THEN 151
140  Top$="N"
150  GOTO 240
151  PRINTER IS S        ! Check to see if printer is present
152  PRINT
153  PRINTER IS 16
160  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
170  PRINT "has top-of-form generation and whether or not you wish to
make"
180  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
190  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
200  PRINT "generation, enter an N and press CONT."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 210
240  PRINT PAGE
250  PRINT "Below you are asked to enter the lower and upper row and c
olumn"
260  PRINT "subscripts of the array you wish to sort.  The subscripts
may be"
270  PRINT "positive, negative, or both.  The only restrictions are th
at the"
280  PRINT "lower row subscript must be less than or equal to the uppe
r row"
290  PRINT "subscript, and the lower column subscript must be less tha
n or"
```

```
300  PRINT "equal to the upper column subscript."
310  INPUT "Enter the lower row subscript",N1
320  INPUT "Enter the upper row subscript",N2
330  IF N1<=N2 THEN 380
340  BEEP
350  DISP "ILLEGAL RANGE"
360  WAIT 700
370  GOTO 310
380  INPUT "Enter the lower column subscript",M1
390  INPUT "Enter the upper column subscript",M2
400  IF M1<=M2 THEN 450
410  BEEP
420  DISP "ILLEGAL RANGE"
430  WAIT 700
440  GOTO 380
450  CALL Rcdriver(N1,N2,M1,M2,S,Top$)
451  BEEP
452  DISP "PROGRAM COMPLETED"
460 END
470 SUB Rcdriver(N1,N2,M1,M2,S,Top$)
480 DIM A(N1:N2,M1:M2),B(N1:N2,M1:M2)
490 RANDOMIZE
500       PRINT PAGE,"If you wish to enter your own array, enter Y in r
esponse"
510       PRINT "to the question below.  If you don't wish to enter you
r own"
520       PRINT "array, but would rather have the machine generate a ra
ndom array,"
530       PRINT "enter N.  If you press CONT without entering a number,
 Y"
540       PRINT "will be assumed as the response."
550       S$="Y"
560       INPUT "Enter your own matrix (Y/N)?",S$
570       IF (S$="Y") OR (S$="y") THEN Manual
580       IF (S$="N") OR (S$="n") THEN Auto
590       GOTO 550
600 Auto: FOR I=N1 TO N2
610        FOR J=M1 TO M2
620        A(I,J)=INT(1000*RND)
630        NEXT J
640        NEXT I
650        GOTO Run
660 Manual: PRINT PAGE
670          PRINTER IS S
680          PRINT "      ";
690          FOR I=M1 TO M2
700          PRINT USING 710;I
```

```
710          IMAGE #,"     "MDD"    "
720          NEXT I
730          PRINT
740          FOR I=N1 TO N2
750          PRINT USING 760;I
760          IMAGE #," "MDD"  "
770          FOR J=M1 TO M2
780          DISP "ELEMENT #";I;J;
790          INPUT A(I,J)
800          PRINT USING 810;A(I,J)
810          IMAGE #,MD.2DE" "
820          NEXT J
830          PRINT
840          NEXT I
850       PRINTER IS 16
860       IF S<>16 THEN PRINT PAGE
870       PRINT LIN(1),"If you want to make any changes, enter the subs
cripts of the"
880       PRINT "element you wish to change.  Otherwise, press CONT"
890       PRINT "without entering anything.",LIN(2)
900       PRINTER IS S
910       I=J=3.141592654
920       INPUT "Enter the subscripts",I,J
930          IF (I=3.141592654) AND (J=3.141592654) THEN Run
940          IF (I>=N1) AND (I<=N2) AND (J>=M1) AND (J<=M2) THEN 990
950          BEEP
960          DISP "ILLEGAL SUBSCRIPTS"
970          WAIT 700
980          GOTO 910
990          DISP "Element #("&VAL$(I)&","&VAL$(J)&")";
1000         INPUT A(I,J)
1010         PRINT "Element #("&VAL$(I)&","&VAL$(J)&") IS "&VAL$(A(I,J)
)
1020         GOTO 910
1030 Run: PRINTER IS S
1040      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE;
1050      PRINT LIN(2),"UNSORTED ARRAY"
1060      CALL Output(A(*),N1,N2,M1,M2)
1070      PRINTER IS 16
1080      IF S<>16 THEN PRINT PAGE;
1090      IF S=16 THEN PRINT LIN(2);
1100      PRINT "     Below, you are asked to decide whether you want t
o sort"
1110      PRINT "the array by row and carry the columns, or by columns
and"
1120      PRINT "carry the rows. Respond with R for rows or C for colum
ns."
```

```
1130      PRINT "if you press CONT without entering anything, R will be
"
1140      PRINT "the assumed response.",LIN(2)
1150      PRINT "      Next you will be asked to select which row or col
umn"
1160      PRINT "you wish to sort on.  Please respond with the proper"
1170      PRINT "subscript.  Out of range subscripts will be rejected.
 There"
1180      PRINT "will be no default subscript, so you must enter a numb
er"
1190      PRINT "before pressing CONT."
1200      S$="R"
1210      W=3.141592654
1220      INPUT "Sort on row or column (R/C)?",S$
1230      IF (S$="R") OR (S$="r") THEN 1270
1240      IF (S$="C") OR (S$="c") THEN 1340
1250      S$="R"
1260      GOTO 1220
1270      C1=1
1280      INPUT "Which row?",W
1290      IF (W>=N1) AND (W<=N2) AND (W<>3.141592654) THEN 1410
1300      BEEP
1310      DISP "ILLEGAL RESPONSE"
1320      WAIT 700
1330      GOTO 1280
1340      C1=0
1350      INPUT "Which column?",W
1360      IF (W>=M1) OR (W<=M2) OR (W<>3.141592654) THEN 1410
1370      BEEP
1380      DISP "ILLEGAL RESPONSE"
1390      WAIT 700
1400      GOTO 1350
1410      IF S<>16 THEN PRINT PAGE;
1420      IF S=16 THEN PRINT LIN(2);
1430      PRINT "Do you want to sort in ascending order or descending o
rder?"
1440      PRINT "If you want to sort in ascending order, enter A.  If y
ou want"
1450      PRINT "to sort in descending order, enter D.  If you press CO
NT"
1460      PRINT "without entering anything, A will be the assumed respo
nse."
1470      S$="A"
1480      INPUT "Do you want to sort in ascending or descending order (
A/D)?",S$
1490      IF (S$<>"A") AND (S$<>"a") AND (S$<>"D") AND (S$<>"d") THEN 1
470
```

```
1500      PRINT PAGE
1510      PRINTER IS S
1520      Incdec=1
1530      IF (S$="d") OR (S$="D") THEN Incdec=0
1540      MAT B=A
1550      DISP "SORTING"
1560      CALL Arraysort_q(A(*),N1,N2,M1,M2,C1,W,Incdec)
1570      DISP ""
1580      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE;
1590      PRINT LIN(2),"SORTED ARRAY"
1600      CALL Output(A(*),N1,N2,M1,M2)
1610      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE
1620      INPUT "Would you like to sort the array differently (Y/N)?",A
nswer$
1630      IF UPC$(Answer$)="Y" THEN 1660
1640      IF UPC$(Answer$)="N" THEN 1700
1650      GOTO 1620
1660      MAT A=B
1670      PRINT LIN(2),"ORIGINAL ARRAY"
1680      CALL Output(A(*),N1,N2,M1,M2)
1690      GOTO 850
1700      PRINTER IS 16
1710 SUBEXIT
1720 SUB Output(A(*),N1,N2,M1,M2)
1730       PRINT "        ";
1740      Linenumber=1
1750      FOR I=M1 TO M2
1760         IF (I+Linenumber-M1) MOD 8<>0 THEN 1790
1770         Linenumber=Linenumber+1
1780         PRINT LIN(1),"         ";
1790         PRINT USING 1800;I
1800         IMAGE #,3XM4D2X
1810      NEXT I
1820      PRINT LIN(1),"         ";
1830      FOR I=M1 TO M2
1840         PRINT USING 1850
1850         IMAGE #,10("-")
1860         IF I+1-M1>=7 THEN 1880
1870      NEXT I
1880      PRINT
1890      FOR I=N1 TO N2
1900         PRINT USING 1910;I
1910         IMAGE #,M4D" | "
1920         Linenumber=1
1930         FOR J=M1 TO M2
1940            IF (J+Linenumber-M1) MOD 8<>0 THEN 1970
1950            Linenumber=Linenumber+1
```

```
1960            PRINT LIN(1),"      | ";
1970            PRINT USING 1980;A(I,J)
1980            IMAGE #,MD.2DEX
1990          NEXT J
2000          PRINT
2010       NEXT I
2020 SUBEXIT
```

Subprogram Name: Arraysort_s

This subprogram allows the user to sort a two-dimensional array on any row or any column into ascending or descending order.

This sort is much slower than the Fast Sort of a Numeric Array on page 47 , but it is more economical with memory. This is for two reasons: 1) The subprogram itself is much shorter and 2) this subprogram requires only one auxiliary storage array, while the Fast Sort requires four auxiliary storage arrays.

Subprogram Utilization:

File Name: SSORTA
Calling Syntax: CALL Arraysort_s (A(*), A,B,C,D,E,F,G)

Input Parameters:

A(*) - The first parameter in the parameter list must be a two-dimensional full-precision numeric array. This is the array which will be sorted.

A - The second parameter must be a full-precision variable, constant, or expression. A is the lower row bound of the sort.

B - The third parameter must be a full-precision variable, constant, or expression. B is the upper row bound of the sort.

C - The fourth parameter must be a full-precision variable, constant, or expression. C is the lower column bound of the sort.

D - The fifth parameter must be a full-precision variable, constant, or expression. D is the upper column bound of the sort.

E - The sixth parameter must be a full-precision variable, constant, or expression. E tells whether the array

A(*) is to be sorted by row or column. If E is 1, then the array will be sorted by row. If E is Ø, then the array will be sorted by column. If E is anything else, no sorting will be done.

F - The seventh parameter must be a full-precision variable, constant, or expression. F tells which row or column is to be sorted.

G - The eighth parameter must be a full-precision variable, constant, or expression. G tells whether the array A(*) is to be sorted in increasing or decreasing order. If G is 1, then the array will be sorted in increasing order. If G is Ø, then A(*) will be sorted in decreasing order. If G is anything else, then no sorting will be done.

Output Parameters:

A(*) - After this subprogram is executed, A(*) (a full-precision two-dimensional array) will be sorted between columns C and D and between rows A and B (see the section under Input Parameters for explanations of A, B, C, and D).

Local Variables:

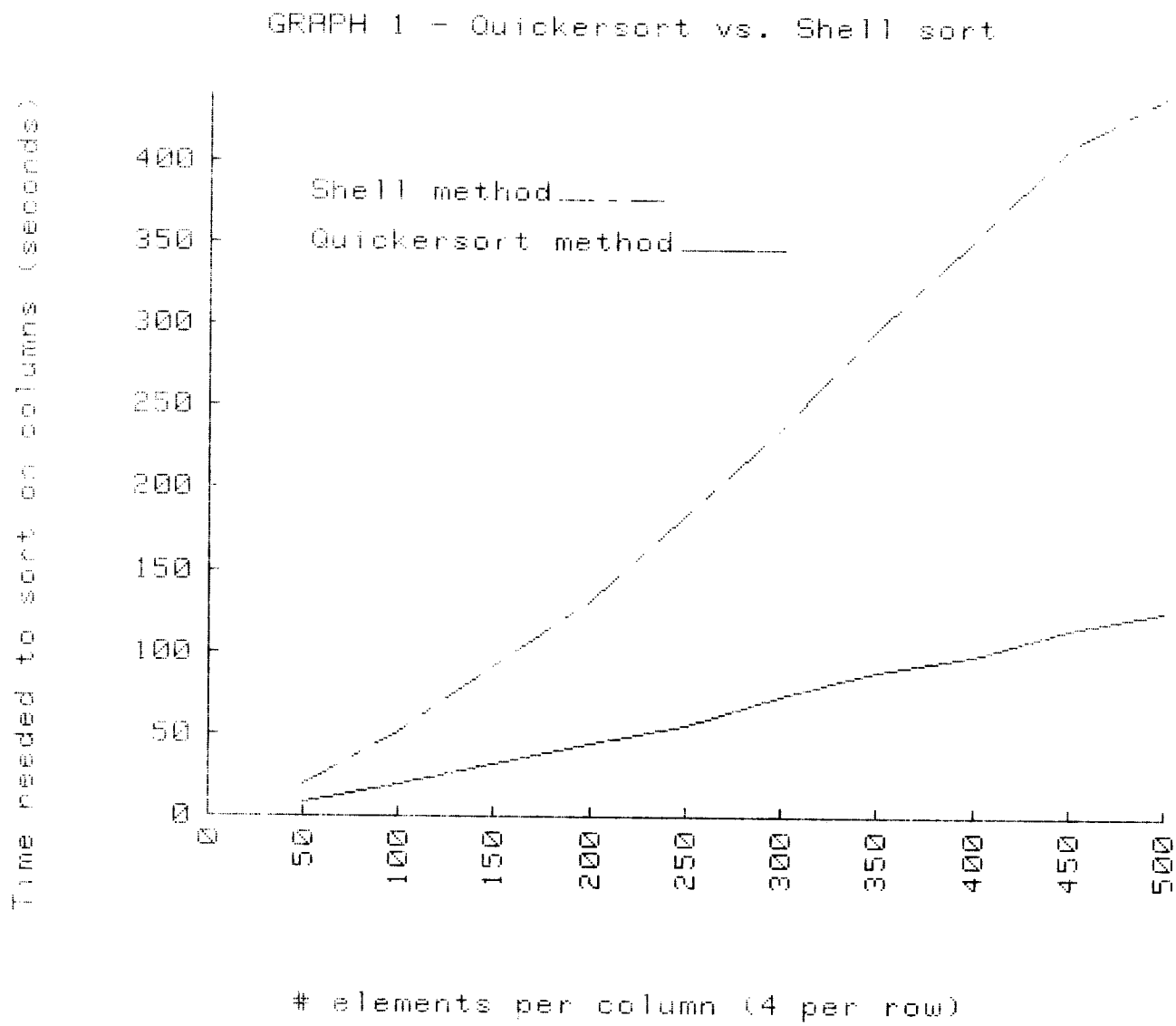
Count - Loop counter
H - $2^{(s-2)}$
I - Temporary subscript
J - Loop counter
S - Counter on loop to sort every $2^{(s-1)}$ element
T - $LOG_2$ of the number of elements to be sorted.
Temp - Temporary subscript variable

Special Considerations and Programming Hints:

1. If the lower row bound of the sort is greater than the upper row bound, or if the lower column bound is greater than the upper column bound then the array will be unchanged from its input state.
2. As mentioned in the introduction, this sort is slower than the Fast Sort on page 47 , but it also uses less memory. The

Fast Sort is recommended unless a memory overflow occurs, in which case it may be necessary to fall back on this sub-program. (Note: See the timing comparison graph immediately following this section - GRAPH 1.)

3. System Configuration:
   Standard Memory Option
   (Optional) Printer


GRAPH 1 - Quickersort vs. Shell sort
Time needed to sort 50 columns (seconds)
400
350
300
250
200
150
100
50
0
Shell method
Quickersort method
0
50
100
150
200
250
300
350
400
450
500
# elements per column (4 per row)

## Annotated Listing of Subprogram Arraysort_s

```
10    SUB Arraysort_s(A(*),N1,N2,M1,M2,C1,W,Incdec)
20    IF C1=1 THEN Row
30    IF C1=0 THEN Column
40    SUBEXIT
50 Row:    CALL Srcsort(A(*),M1,M2,N1,N2,C1,W,Incdec)
60    SUBEXIT
70 Column:    CALL Srcsort(A(*),N1,N2,M1,M2,C1,W,Incdec)
80    SUBEXIT
90    SUB Srcsort(A(*),Sortlow,Sorthigh,Carrylow,Carryhigh,C1,W,Incdec)
100   DIM K(Carrylow:Carryhigh)
110   INTEGER Count,S,J
120   IF (Incdec<>0) AND (Incdec<>1) OR (Sortlow>=Sorthigh) OR (Carrylo
w>Carryhigh) THEN SUBEXIT
121   IF (W<Carrylow) OR (W>Carryhigh) THEN SUBEXIT
130   T=INT(LOG(Sorthigh+1-Sortlow)/LOG(2))+1
140   FOR S=T TO 1 STEP -1
150      H=2^(S-1)
160      FOR J=H+Sortlow TO Sorthigh
170         I=J-H
180         FOR Count=Carrylow TO Carryhigh
190         K(Count)=A(C1*Count+NOT C1*J,NOT C1*Count+C1*J)
200         NEXT Count
210 Decide:   IF Incdec=0 THEN Decrease
220 Increase:   IF K(W)>=A(C1*W+NOT C1*I,NOT C1*W+C1*I) THEN Insert
230             GOTO Switch
240 Decrease:   IF K(W)<=A(C1*W+NOT C1*I,NOT C1*W+C1*I) THEN Insert
250 Switch:     Temp=I+H
260             FOR Count=Carrylow TO Carryhigh
270             A(C1*Count+NOT C1*Temp,NOT C1*Count+C1*Temp)=A(C1*Count
+NOT C1*I,NOT C1*Count+C1*I)
280             NEXT Count
290             I=I-H
300             IF I>=Sortlow THEN Decide
310 Insert:     Temp=I+H
320             FOR Count=Carrylow TO Carryhigh
330             A(C1*Count+NOT C1*Temp,NOT C1*Count+C1*Temp)=K(Count)
340             NEXT Count
350          NEXT J
360       NEXT S
370       SUBEXIT
```

Method:

The method used in this subprogram is Donald Shell's dimishing increment sort. First, a sequence of diminishing increments $h_t$, $h_{t-1}$,..., $h_1$ is selected. (In this case, t is $\log_2$ N, where N is the number of elements to be sorted, and the h's are $2^{t-1}$, $2^{t-2}$, .... $2^0$.) Then, for each h, a straight insertion sort is performed on those elements of the array which are h elements apart. By the time the last h has been used the array is sorted. (Note: The other h's may be anything, as long as they are diminishing, but $h_1$ must <u>always</u> be 1.)

REFERENCE:

Knuth, Donald E., <u>The Art of Computer Programming</u> Vol. 3 (<u>Sorting and Searching</u>) (Addison-Wesley 1973) pp. 84-85.

Driver Utilization:

File Name: SSRTAD

This driver allows the user to either enter an array of his own, or have a random array generated. The array will then be printed, and the user will be given a choice of sorting either by row or by column, and in either descending or ascending order. The driver will then call the subprogram Arraysort_s which will sort the array. Upon returning to the driver, the sorted array will be printed and titled.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SSRTAD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.

         or
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT

c) Go to step 5.

5. When "Enter the lower row subscript" appears in the display area:
   a. Enter: The lower row boundary of the array.
   b. Press: CONT
6. When "Enter the upper row subscript" appears in the display area:
   a. Enter: The upper row boundary of the array.
   b. Press: CONT
7. When "Enter the lower column subscript" appears in the display area:
   a. Enter: The lower column boundary.
   b. Press: CONT
8. When "Enter the upper column subscript" appears in the display area:
   a. Enter: The upper column boundary of the array.
   b. Press: CONT
9. When "Enter your own matrix (Y/N)?" appears in the display area:
   a. If you wish to enter your own matrix:
      1) Enter: Y
      2) Press: CONT
      3) Go to step 10.

      or

   a. If you do not wish to enter your own matrix, but would rather have the machine generate one for you:
      1) Enter: N
      2) Press: CONT
      3) Go to step 12.
10. When "ELEMENT # i, j?" appears in the display area:
   a. Enter: The matrix element belonging in the jth column of the ith row.
   b. Press: CONT
   c. Repeat step 10 as often as necessary.
11. When "Enter the subscripts" appears in the display area:
   a. If you wish to make any changes in the matrix:
      1) Enter: The subscripts of the element you wish to change

2) Press: CONT

3) Go to step 10.

or

a. If you do not wish to make any changes in the matrix:

1) Press: CONT

2) Go to step 12.

12. When "Sort by row or column (R/C)?" appears in the display area:

a. If you want to sort by row:

1) Enter: R

2) Press: CONT

b. When "Which row?" appears in the display area:

1) Enter: The row number you want to sort on.

2) Press: CONT

or

a. If you want to sort by column:

1) Enter: C

2) Press: CONT

b. When "Which column?" appears in the display area:

1) Enter: The column number you want to sort on.

2) Press: CONT

13. When "Do you want to sort in ascending or descending order (A/D)?" appears in the display area:

a. If you wish to sort in ascending order:

1) Enter: A

2) Press: CONT

or

a. If you wish to sort in descending order:

1) Enter: D

2) Press: CONT

14. The sorted array will be printed and titled.

15. When "Would you like to sort the array differently (Y/N)?" appears in the display area:

a. If you would like to sort the array differently:

1) Type: Y

2) Press: CONT

3) Go to step 11.

or

a. If you do not want to sort the array again:

1) Type: N

2) Press: CONT

3) The program stops.

UNSORTED ARRAY

| | -28 | -27 | -26 | -25 | -24 |
|---|---|---|---|---|---|
| -15 | 6.42E+02 | 8.10E+01 | 8.51E+02 | 9.83E+02 | 9.90E+02 |
| -14 | 2.41E+02 | 4.83E+02 | 4.56E+02 | 9.94E+02 | 4.34E+02 |
| -13 | 4.63E+02 | 8.60E+01 | 4.92E+02 | 5.25E+02 | 5.90E+01 |
| -12 | 6.71E+02 | 4.78E+02 | 7.90E+02 | 9.79E+02 | 4.90E+01 |
| -11 | 6.68E+02 | 9.77E+02 | 7.43E+02 | 9.70E+02 | 1.99E+02 |
| -10 | 7.34E+02 | 1.21E+02 | 5.94E+02 | 7.11E+02 | 4.02E+02 |
| -9 | 6.51E+02 | 7.24E+02 | 1.45E+02 | 5.29E+02 | 3.17E+02 |
| -8 | 8.62E+02 | 4.92E+02 | 5.29E+02 | 7.30E+01 | 5.16E+02 |
| -7 | 6.97E+02 | 6.95E+02 | 9.43E+02 | 7.46E+02 | 4.10E+02 |
| -6 | 5.38E+02 | 8.16E+02 | 5.81E+02 | 1.57E+02 | 1.07E+02 |
| -5 | 2.56E+02 | 6.59E+02 | 3.25E+02 | 4.48E+02 | 9.98E+02 |
| -4 | 6.22E+02 | 6.30E+02 | 5.35E+02 | 0.00E+00 | 1.21E+02 |
| -3 | 4.11E+02 | 4.53E+02 | 7.89E+02 | 2.20E+01 | 8.87E+02 |
| -2 | 1.07E+02 | 7.09E+02 | 5.53E+02 | 7.48E+02 | 4.40E+02 |
| -1 | 1.16E+02 | 3.87E+02 | 8.24E+02 | 4.90E+01 | 9.33E+02 |
| 0 | 3.27E+02 | 2.79E+02 | 6.68E+02 | 9.23E+02 | 4.50E+01 |
| 1 | 4.68E+02 | 2.52E+02 | 1.23E+02 | 5.59E+02 | 4.83E+02 |
| 2 | 8.02E+02 | 2.76E+02 | 1.41E+02 | 7.47E+02 | 4.42E+02 |
| 3 | 1.62E+02 | 5.32E+02 | 8.20E+01 | 6.21E+02 | 4.75E+02 |
| 4 | 6.96E+02 | 9.84E+02 | 6.01E+02 | 6.56E+02 | 9.40E+02 |
| 5 | 8.08E+02 | 8.91E+02 | 2.20E+01 | 2.52E+02 | 1.35E+02 |
| 6 | 1.60E+01 | 9.85E+02 | 9.29E+02 | 6.92E+02 | 4.39E+02 |
| 7 | 2.35E+02 | 4.05E+02 | 7.76E+02 | 3.93E+02 | 7.60E+02 |
| 8 | 3.28E+02 | 6.18E+02 | 2.49E+02 | 9.24E+02 | 8.83E+02 |
| 9 | 8.60E+02 | 3.03E+02 | 3.46E+02 | 8.20E+02 | 5.10E+01 |
| 10 | 3.00E+01 | 1.98E+02 | 1.38E+02 | 2.28E+02 | 4.64E+02 |
| 11 | 8.12E+02 | 4.80E+01 | 5.82E+02 | 1.23E+02 | 1.55E+02 |
| 12 | 3.85E+02 | 8.02E+02 | 7.18E+02 | 3.50E+01 | 5.85E+02 |
| 13 | 7.09E+02 | 9.68E+02 | 1.61E+02 | 1.10E+02 | 5.99E+02 |
| 14 | 7.79E+02 | 3.03E+02 | 2.88E+02 | 5.07E+02 | 5.90E+01 |
| 15 | 6.76E+02 | 8.70E+02 | 1.71E+02 | 9.09E+02 | 8.77E+02 |

The array is sorted on column -27 in ascending order

SORTED ARRAY

| | -28 | -27 | -26 | -25 | -24 |
|---|---|---|---|---|---|
| -15 | 3.12E+02 | 4.80E+01 | 5.82E+02 | 1.23E+02 | 1.55E+02 |
| -14 | 6.42E+02 | 8.10E+01 | 8.51E+02 | 9.83E+02 | 9.90E+02 |
| -13 | 4.63E+02 | 8.60E+01 | 4.92E+02 | 5.25E+02 | 5.90E+01 |
| -12 | 7.34E+02 | 1.21E+02 | 5.94E+02 | 7.11E+02 | 4.02E+02 |
| -11 | 2.00E+01 | 1.98E+02 | 1.38E+02 | 2.28E+02 | 4.64E+02 |
| -10 | 4.68E+02 | 2.52E+02 | 1.23E+02 | 5.59E+02 | 4.83E+02 |
| -9 | 8.02E+02 | 2.76E+02 | 1.41E+02 | 7.47E+02 | 4.42E+02 |
| -8 | 3.27E+02 | 2.79E+02 | 6.68E+02 | 9.23E+02 | 4.50E+01 |
| -7 | 8.60E+02 | 3.03E+02 | 3.46E+02 | 8.20E+02 | 5.10E+01 |
| -6 | 7.79E+02 | 3.03E+02 | 2.88E+02 | 5.07E+02 | 5.90E+01 |
| -5 | 1.16E+02 | 3.87E+02 | 8.24E+02 | 4.90E+01 | 9.33E+02 |
| -4 | 2.35E+02 | 4.05E+02 | 7.76E+02 | 3.93E+02 | 7.60E+02 |
| -3 | 4.11E+02 | 4.53E+02 | 7.89E+02 | 2.20E+01 | 8.87E+02 |
| -2 | 6.71E+02 | 4.78E+02 | 7.90E+02 | 9.79E+02 | 4.90E+01 |
| -1 | 2.41E+02 | 4.83E+02 | 4.56E+02 | 9.94E+02 | 4.34E+02 |
| 0 | 8.62E+02 | 4.92E+02 | 5.29E+02 | 7.30E+01 | 5.16E+02 |
| 1 | 1.62E+02 | 5.32E+02 | 8.20E+01 | 6.21E+02 | 4.75E+02 |
| 2 | 3.28E+02 | 6.18E+02 | 2.49E+02 | 9.24E+02 | 8.83E+02 |
| 3 | 6.22E+02 | 6.30E+02 | 5.35E+02 | 0.00E+00 | 1.21E+02 |
| 4 | 2.56E+02 | 6.59E+02 | 3.25E+02 | 4.48E+02 | 9.98E+02 |
| 5 | 6.97E+02 | 6.95E+02 | 9.43E+02 | 7.46E+02 | 4.10E+02 |
| 6 | 1.07E+02 | 7.09E+02 | 5.53E+02 | 7.48E+02 | 4.40E+02 |
| 7 | 6.51E+02 | 7.24E+02 | 1.45E+02 | 5.29E+02 | 3.17E+02 |
| 8 | 3.85E+02 | 8.02E+02 | 7.18E+02 | 3.50E+01 | 5.85E+02 |
| 9 | 5.38E+02 | 8.16E+02 | 5.81E+02 | 1.57E+02 | 1.07E+02 |
| 10 | 6.76E+02 | 8.70E+02 | 1.71E+02 | 9.09E+02 | 8.77E+02 |
| 11 | 8.08E+02 | 8.91E+02 | 2.20E+01 | 2.52E+02 | 1.35E+02 |
| 12 | 7.09E+02 | 9.68E+02 | 1.61E+02 | 1.10E+02 | 5.99E+02 |
| 13 | 6.68E+02 | 9.77E+02 | 7.43E+02 | 9.70E+02 | 1.99E+02 |
| 14 | 6.96E+02 | 9.84E+02 | 6.01E+02 | 6.56E+02 | 9.40E+02 |
| 15 | 1.60E+01 | 9.85E+02 | 9.29E+02 | 6.92E+02 | 4.39E+02 |

# Driver Listing

```
10   LINK "SSORTA",2030
20   PRINTER IS 16
30   PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is"
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONT without entering"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90   S=16
100  INPUT "Printer select code?",S
110  S=INT(S)
120  IF (S>16) OR (S<0) THEN 90
130  IF S<>16 THEN 151
140  Top$="N"
150  GOTO 240
151  PRINTER IS S
152  PRINT
153  PRINTER IS 16
160  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
170  PRINT "has top-of-form generation and whether or not you wish to
make"
180  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
190  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
200  PRINT "generation, enter an N and press CONT."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 210
240  PRINT PAGE
250  PRINT "Below you are asked to enter the lower and upper row and c
olumn"
260  PRINT "subscripts of the array you wish to sort.  The subscripts
may be"
270  PRINT "positive, negative, or both.  The only restrictions are th
at the"
280  PRINT "lower row subscript must be less than or equal to the uppe
r row"
290  PRINT "subscript, and the lower column subscript must be less tha
n or"
```

```
300  PRINT "equal to the upper column subscript."
310  INPUT "Enter the lower row subscript",N1
320  INPUT "Enter the upper row subscript",N2
330  IF N1<=N2 THEN 380
340  BEEP
350  DISP "ILLEGAL RANGE"
360  WAIT 700
370  GOTO 310
380  INPUT "Enter the lower column subscript",M1
390  INPUT "Enter the upper column subscript",M2
400  IF M1<=M2 THEN 450
410  BEEP
420  DISP "ILLEGAL RANGE"
430  WAIT 700
440  GOTO 380
450  CALL Rcdriver(N1,N2,M1,M2,S,Top$)
451  BEEP
452  DISP "PROGRAM COMPLETED"
460 END
470 SUB Rcdriver(N1,N2,M1,M2,S,Top$)
480 DIM A(N1:N2,M1:M2),B(N1:N2,M1:M2)
490 RANDOMIZE
500       PRINT PAGE,"If you wish to enter your own array, enter Y in r
esponse"
510       PRINT "to the question below.  If you don't wish to enter you
r own"
520       PRINT "array, but would rather have the machine generate a ra
ndom array,"
530       PRINT "enter N.  If you press CONT without entering a number,
 Y"
540       PRINT "will be assumed as the response."
550       S$="Y"
560       INPUT "Enter your own matrix (Y/N)?",S$
570       IF (S$="Y") OR (S$="y") THEN Manual
580       IF (S$="N") OR (S$="n") THEN Auto
590       GOTO 550
600 Auto: FOR I=N1 TO N2
610        FOR J=M1 TO M2
620        A(I,J)=INT(1000*RND)
630        NEXT J
640        NEXT I
650        GOTO Run
660 Manual: PRINT PAGE
670          PRINTER IS S
680          PRINT "      ";
690          FOR I=M1 TO M2
700          PRINT USING 710;I
```

```
710          IMAGE #,"    "MDD"    "
720          NEXT I
730          PRINT
740          FOR I=N1 TO N2
750          PRINT USING 760;I
760          IMAGE #," "MDD"  "
770          FOR J=M1 TO M2
780          DISP "ELEMENT #";I;J;
790          INPUT A(I,J)
800          PRINT USING 810;A(I,J)
810          IMAGE #,MD.2DE" "
820          NEXT J
830          PRINT
840          NEXT I
850       PRINTER IS 16
860       IF S<>16 THEN PRINT PAGE
870       PRINT LIN(1),"If you want to make any changes, enter the subs
cripts of the"
880       PRINT "element you wish to change.  Otherwise, press CONT"
890       PRINT "without entering anything.",LIN(2)
900       PRINTER IS S
910       I=J=3.141592654
920       INPUT "Enter the subscripts",I,J
930          IF (I=3.141592654) AND (J=3.141592654) THEN Run
940          IF (I>=N1) AND (I<=N2) AND (J>=M1) AND (J<=M2) THEN 990
950          BEEP
960          DISP "ILLEGAL SUBSCRIPTS"
970          WAIT 700
980          GOTO 910
990          DISP "Element #("&VAL$(I)&","&VAL$(J)&")";
1000         INPUT A(I,J)
1010         PRINT "Element #("&VAL$(I)&","&VAL$(J)&") IS "&VAL$(A(I,J)
)
1020         GOTO 910
1030 Run: PRINTER IS S
1040      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE;
1050      PRINT LIN(2),"UNSORTED ARRAY"
1060      CALL Output(A(*),N1,N2,M1,M2)
1070      PRINTER IS 16
1080      IF S<>16 THEN PRINT PAGE;
1090      IF S=16 THEN PRINT LIN(2);
1100      PRINT "     Below, you are asked to decide whether you want t
o sort"
1110      PRINT "the array by row and carry the columns, or by columns
and"
1120      PRINT "carry the rows. Respond with R for rows or C for colum
ns."
```

```
1130      PRINT "if you press CONT without entering anything, R will be
"
1140      PRINT "the assumed response.",LIN(2)
1150      PRINT "      Next you will be asked to select which row or col
umn"
1160      PRINT "you wish to sort on.  Please respond with the proper"
1170      PRINT "subscript.  Out of range subscripts will be rejected.
 There"
1180      PRINT "will be no default subscript, so you must enter a numb
er"
1190      PRINT "before pressing CONT."
1200      S$="R"
1210      W=3.141592654
1220      INPUT "Sort on row or column (R/C)?",S$
1230      IF (S$="R") OR (S$="r") THEN 1270
1240      IF (S$="C") OR (S$="c") THEN 1340
1250      S$="R"
1260      GOTO 1220
1270      C1=1
1280      INPUT "Which row?",W
1290      IF (W>=N1) AND (W<=N2) AND (W<>3.141592654) THEN 1410
1300      BEEP
1310      DISP "ILLEGAL RESPONSE"
1320      WAIT 700
1330      GOTO 1280
1340      C1=0
1350      INPUT "Which column?",W
1360      IF (W>=M1) OR (W<=M2) OR (W<>3.141592654) THEN 1410
1370      BEEP
1380      DISP "ILLEGAL RESPONSE"
1390      WAIT 700
1400      GOTO 1350
1410      IF S<>16 THEN PRINT PAGE;
1420      IF S=16 THEN PRINT LIN(2);
1430      PRINT "Do you want to sort in ascending order or descending o
rder?"
1440      PRINT "If you want to sort in ascending order, enter A.  If y
ou want"
1450      PRINT "to sort in descending order, enter D.  If you press CO
NT"
1460      PRINT "without entering anything, A will be the assumed respo
nse."
1470      S$="A"
1480      INPUT "Do you want to sort in ascending or descending order (
A/D)?",S$
1490      IF (S$<>"A") AND (S$<>"a") AND (S$<>"D") AND (S$<>"d") THEN 1
470
```

```
1500      PRINT PAGE
1510      PRINTER IS S
1520      Incdec=1
1530      IF (S$="d") OR (S$="D") THEN Incdec=0
1540      MAT B=A
1550      DISP "SORTING"
1560      CALL Arraysort_s(A(*),N1,N2,M1,M2,C1,W,Incdec)
1570      DISP ""
1580      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE;
1590      PRINT LIN(2),"SORTED ARRAY"
1600      CALL Output(A(*),N1,N2,M1,M2)
1610      IF (S<>16) AND ((Top$="y") OR (Top$="Y")) THEN PRINT PAGE
1620      INPUT "Would you like to sort the array differently (Y/N)?",A
nswer$
1630      IF UPC$(Answer$)="Y" THEN 1660
1640      IF UPC$(Answer$)="N" THEN 1700
1650      GOTO 1620
1660      MAT A=B
1670      PRINT LIN(2),"ORIGINAL ARRAY"
1680      CALL Output(A(*),N1,N2,M1,M2)
1690      GOTO 850
1700      PRINTER IS 16
1710 SUBEXIT
1720 SUB Output(A(*),N1,N2,M1,M2)
1730       PRINT "         ";
1740       Linenumber=1
1750       FOR I=M1 TO M2
1760          IF (I+Linenumber-M1) MOD 8<>0 THEN 1790
1770          Linenumber=Linenumber+1
1780          PRINT LIN(1),"         ";
1790          PRINT USING 1800;I
1800          IMAGE #,3X,M4D,2X
1810       NEXT I
1820       PRINT LIN(1),"         ";
1830       FOR I=M1 TO M2
1840          PRINT USING 1850
1850          IMAGE #,10("-")
1860          IF I+1-M1>=7 THEN 1880
1870       NEXT I
1880       PRINT
1890       FOR I=N1 TO N2
1900          PRINT USING 1910;I
1910          IMAGE #,M4D," | "
1920          Linenumber=1
1930          FOR J=M1 TO M2
1940             IF (J+Linenumber-M1) MOD 8<>0 THEN 1970
1950             Linenumber=Linenumber+1
```

```
1960             PRINT LIN(1),"      | ";
1970             PRINT USING 1980;A(I,J)
1980             IMAGE #,MD.2DEX
1990          NEXT J
2000          PRINT
2010       NEXT I
2020 SUBEXIT
```

HEWLETT hp PACKARD

# FAST SORT OF A STRING VECTOR

Subprogram Name: Stringsort_q

This program allows the user to sort a one-dimensional string vector into either ascending or descending order.

This sort is faster than the Minimal Storage Sort of a String Vector on page 107 but it also requires more memory. The subprogram itself needs 2788 bytes more memory, and there are two auxiliary storage arrays.

Subprogram Utilization:

File Name: QSORTS

Calling Syntax: CALL Stringsort_q (A$(*),A,B,C,D)

Input Parameters:

A$(*) - The first parameter in the parameter list must be a one-dimensional string array. This is the string vector which will be sorted.

A - The second parameter must be a full-precision variable, constant, or expression. A represents the lower bound of the sort.

B - The third parameter must be a full-precision variable, constant, or expression. B represents the upper bound of the sort.

C - The fourth parameter must be a full-precision variable, constant, or expression. C represents the length of each member of the string vector.

D - The last parameter must be a full-precision variable, constant, or expression. D indicates whether the sort will be ascending or descending. If D = 0, the sort will be descending. If D = 1, the sort will be ascending. If D is anything else, the vector will not be sorted.

Output Parameters:

A$(*) - After this subprogram is executed, this string vector will be sorted from elements A through B (if the value passed in as D is Ø or 1 (see the section under Input Parameters)).

Local Variables:

I - Lower endpoint of current array segment
J - Upper endpoint of current array segment
K - Temporary storage for I
L - Temporary storage for L
L(*) - Array for lower endpoints of array segments
Log - $Log_2$ of the number of elements being sorted
M - Stack pointer
T$ - Temporary variable for switching elements
T1$ - Temporary
U(*) - Array for upper endpoints of array segments

Special Considerations and Programming Hints:

1. If the lower sort bound is greater than the upper sort bound, the array will remain unchanged from its input state. If the value passed in for the string length (the fourth parameter of the subprogram call) is less than one, an error will occur because a string can not be dimensioned to have a zero or negative length.
2. As mentioned in the introduction, this sort takes more memory than the Minimal Storage String Sort on page 107 but it is much faster. Because of its speed, this subprogram is recommended over the Minimal Storage Sort. However, if a memory overflow occurs while running this subprogram, it may be necessary to use the Minimal Storage String Sort. (Note: See GRAPH 1 immediately following this section for timing comparison.)
3. This sort may be made faster by removing the capability of sorting both in increasing and decreasing order. An extra listing is included showing which lines need to be deleted in order to accomplish this. The modified listing (which may

be found immediately following GRAPH 2) will sort in ascending order. To sort in descending order, merely reverse the inequality comparisons on all statements having the "I" labels (I1, I2, I3, etc.). Thus, statement I1 would become:

```
I1:  IF  A$(I)> = T$  THEN Lowmiddle 1
```

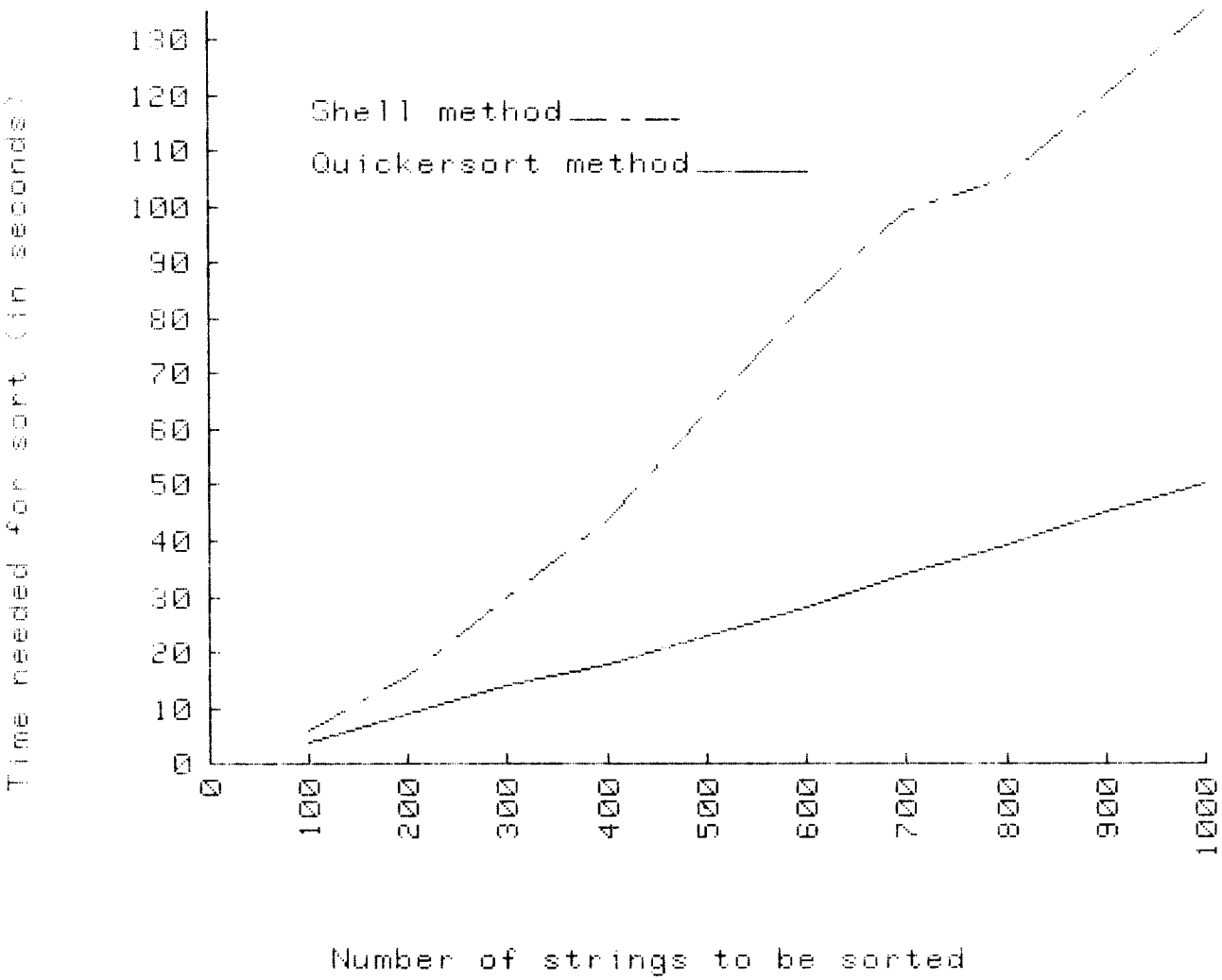
GRAPH 2 immediately following this section shows how much improvement in speed is gained by using the procedure outlined above.

4. System Configuration:

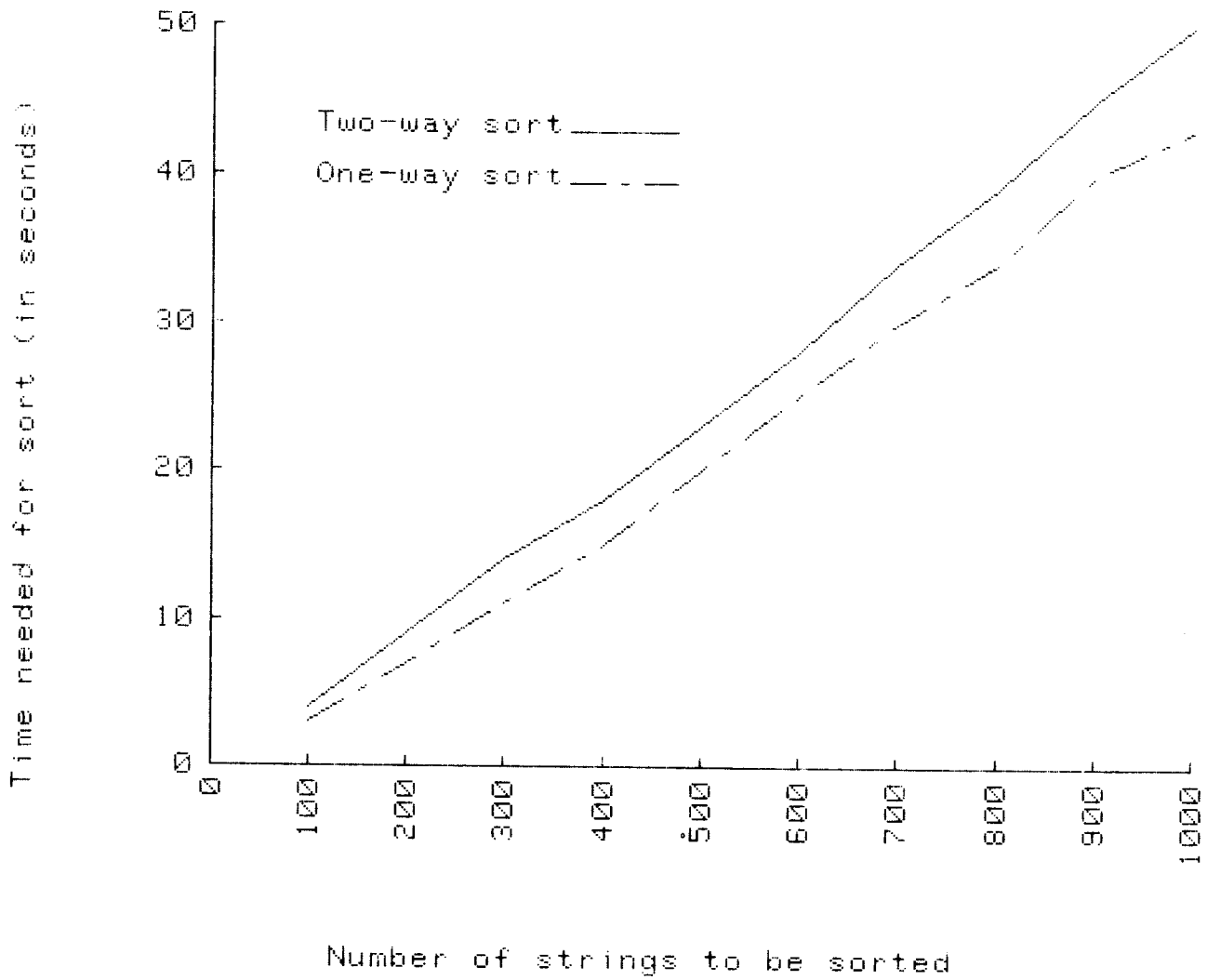
   Standard Memory Option

   (Optional) Printer

GRAPH 1 - Quickersort vs. Shell sort



GRAPH 2 - Two-way sort vs. One way sort



Modified Subprogram Listing (Ascending sort only)

This listing shows those lines which can be removed to make the sort faster. (The deleted lines would allow the user to sort both in ascending and descending order if they were left in.)

```
1010 SUB Stringsort_q(A$(*),I1,J1,Length,Incdec)
1020      N=J1+1-I1
1030      Logtwo=INT(LGT(N)/LGT(2))+1
1040      CALL Qsort(A$(*),Logtwo,I1,J1,Length,Incdec)
1050 SUBEXIT
1060 SUB Qsort(A$(*),Log,I1,J1,Length,Incdec)
1070 OPTION BASE 1
1080 DIM L(Log),U(Log),T$[Length],T1$[Length]
1090      M=1                                  ! Set stack pointer.
1100      I=I1                                 ! Set lower endpoint.
1110      J=J1                                 ! Set upper endpoint.
1120 Start1:IF I>=J THEN Nextgroup
1130 Start2:K=I
1140         I2=INT((J+I)/2)                   ! Determine the midpoint of a
 segment.
1150         T$=A$(I2)
1160         IF Incdec=0 THEN B1
1170 I1:     IF A$(I)<=T$ THEN Lowmiddle1
1180         GOTO 1200
1190 B1:     IF A$(I)>=T$ THEN Lowmiddle1     ! Check to see if lower end
point and
1200         A$(I2)=A$(I)                      !  midpoint are in order.
If not,
1210         A$(I)=T$                          !  switch them.
1220         T$=A$(I2)                         ! Reset midpoint.
1230 Lowmiddle1: L=J                             ! Set upper endpoint.
1240             IF Incdec=0 THEN B2
1250 I2:         IF A$(J)>=T$ THEN Middlehigh
1260             GOTO 1280
1270 B2:         IF A$(J)<=T$ THEN Middlehigh ! Check to see if the mi
dpoint and
1280             A$(I2)=A$(J)                    ! the upper endpoint are
 in order.
1290             A$(J)=T$                        ! If not, switch them.
1300             T$=A$(I2)
1310             IF Incdec=0 THEN B3
1320 I3:         IF A$(I)<=T$ THEN Middlehigh
1330             GOTO 1350
1340 B3:         IF A$(I)>=T$ THEN Middlehigh ! Check to see if the s
witch left
1350             A$(I2)=A$(I)                    !  the lower endpoint a
nd the mid-
1360             A$(I)=T$                        !  point in order.
1370             T$=A$(I2)                       ! If not, switch them.
1380 Middlehigh: L=L-1                           ! Decrement the upper e
ndpoint.
1390             IF Incdec=0 THEN B4
```

```
1400 I4:          IF A$(L)>T$ THEN Middlehigh
1410              GOTO 1430
1420 B4:          IF A$(L)<T$ THEN Middlehigh  ! Check to see if the n
ew upper
1430              T1$=A$(L)                    !  endpoint is in order
.
1440 Stepup: K=K+1                        ! If not, save the upper endpoin
t and
1450         IF Incdec=0 THEN B5
1460 I5:     IF A$(K)<T$ THEN Stepup  !  increment the lower endpoint.
  Now
1470         GOTO 1490
1480 B5:     IF A$(K)>T$ THEN Stepup
1490         IF K>L THEN Passed       !  check if the lower endpoint i
s less
1500         A$(L)=A$(K)              !  than the midpoint.  If not, t
hen switch
1510         A$(K)=T1$                !  the upper and lower endpoints
.
1520         GOTO Middlehigh
1530 Passed: IF L-I<=J-K THEN Storehigh   ! Sort the shortest segment
first.
1540         L(M)=I                        ! Store the lower
1550         U(M)=L                        !  endpoints.
1560         I=K                           ! Set the new lower endpoint
.
1570         M=M+1                         ! Push the stack
1580         GOTO 1630
1590 Storehigh: L(M)=K                     ! Store the upper
1600            U(M)=J                     !  endpoints
1610            J=L                        ! Set the new upper endpoint
.
1620            M=M+1                      ! Push the stack
1630            IF J-I>=11 THEN Start2
1640            IF I=I1 THEN Start1
1650            I=I-1
1660 Inc: I=I+1                      ! Increment lower endpoint.
1670      IF I=J THEN Nextgroup      ! If the current segment is sorted
, then
1680      T$=A$(I+1)                 !  sort the next segment.
1690      IF Incdec=0 THEN B6
1700 I6:  IF A$(I)<=T$ THEN Inc
1710      GOTO 1730
1720 B6:  IF A$(I)>=T$ THEN Inc      ! Check to see if next element is
in order.
1730      K=I                        ! Insert element in otherwise sort
ed list.
```

```
1740 Copy: A$(K+1)=A$(K)              ! This section bumps the array up.
1750       K=K-1                      ! Prepare to bump next element.
1760       IF Incdec=0 THEN B7
1770 I7:   IF T$<A$(K) THEN Copy
1780       GOTO 1800
1790 B7:   IF T$>A$(K) THEN Copy      ! Check to see if insertion is h
ere.
1800       A$(K+1)=T$                 ! If so, then insert.
1810       GOTO Inc
1820 Nextgroup: M=M-1                 ! Pop the stack.
1830            IF M=0 THEN Out       ! Check for end conditions.
1840            I=L(M)                ! Restore the
1850            J=U(M)                !  previous endpoints.
1860            GOTO 1630
1870 Out: SUBEXIT
```

# Annotated Listing of Subprogram Stringsort_q

```
1010 SUB Stringsort_q(A$(*),I1,J1,Length,Incdec)
1020     N=J1+1-I1
1030     Logtwo=INT(LGT(N)/LGT(2))+1
1040     CALL Qsort(A$(*),Logtwo,I1,J1,Length,Incdec)
1050 SUBEXIT
1060 SUB Qsort(A$(*),Log,I1,J1,Length,Incdec)
1070 OPTION BASE 1
1080 DIM L(Log),U(Log),T$[Length],T1$[Length]
1090     M=1                                ! Set stack pointer.
1100     I=I1                               ! Set lower endpoint.
1110     J=J1                               ! Set upper endpoint.
1120 Start1:IF I>=J THEN Nextgroup
1130 Start2:K=I
1140         I2=INT((J+I)/2)                ! Determine the midpoint of a
 segment.
1150         T$=A$(I2)
1160         IF Incdec=0 THEN D1
1170 I1:     IF A$(I)<=T$ THEN Lowmiddle1
1180         GOTO 1200
1190 D1:     IF A$(I)>=T$ THEN Lowmiddle1   ! Check to see if lower end
point and
1200         A$(I2)=A$(I)                     !  midpoint are in order.
If not,
1210         A$(I)=T$                         !  switch them.
1220         T$=A$(I2)                        ! Reset midpoint.
1230 Lowmiddle1: L=J                            ! Set upper endpoint.
1240             IF Incdec=0 THEN D2
1250 I2:         IF A$(J)>=T$ THEN Middlehigh
1260             GOTO 1280
1270 D2:         IF A$(J)<=T$ THEN Middlehigh ! Check to see if the mi
dpoint and
1280             A$(I2)=A$(J)                   ! the upper endpoint are
 in order.
1290             A$(J)=T$                       ! If not, switch them.
1300             T$=A$(I2)
1310             IF Incdec=0 THEN D3
1320 I3:         IF A$(I)<=T$ THEN Middlehigh
1330             GOTO 1350
1340 D3:         IF A$(I)>=T$ THEN Middlehigh  ! Check to see if the s
witch left
1350             A$(I2)=A$(I)                   !  the lower endpoint a
nd the mid-
1360             A$(I)=T$                       !  point in order.
1370             T$=A$(I2)                      ! If not, switch them.
1380 Middlehigh: L=L-1                          ! Decrement the upper e
ndpoint.
1390             IF Incdec=0 THEN D4
```

```
1400 I4:          IF A$(L)>T$ THEN Middlehigh
1410              GOTO 1430
1420 D4:          IF A$(L)<T$ THEN Middlehigh   ! Check to see if the n
ew upper
1430              T1$=A$(L)                     !  endpoint is in order
.
1440 Stepup: K=K+1                        ! If not, save the upper endpoin
t and
1450         IF Incdec=0 THEN D5
1460 I5:     IF A$(K)<T$ THEN Stepup      !  increment the lower endpoint.
  Now
1470         GOTO 1490
1480 D5:     IF A$(K)>T$ THEN Stepup
1490         IF K>L THEN Passed           !  check if the lower endpoint i
s less
1500         A$(L)=A$(K)                  !  than the midpoint.  If not, t
hen switch
1510         A$(K)=T1$                    !  the upper and lower endpoints
.
1520         GOTO Middlehigh
1530 Passed: IF L-I<=J-K THEN Storehigh    ! Sort the shortest segment
first.
1540         L(M)=I                        ! Store the lower
1550         U(M)=L                        !  endpoints.
1560         I=K                           ! Set the new lower endpoint
.
1570         M=M+1                         ! Push the stack
1580         GOTO 1630
1590 Storehigh: L(M)=K                     ! Store the upper
1600            U(M)=J                     !  endpoints
1610            J=L                        ! Set the new upper endpoint
.
1620            M=M+1                      ! Push the stack
1630            IF J-I>=11 THEN Start2
1640            IF I=I1 THEN Start1
1650            I=I-1
1660 Inc: I=I+1                       ! Increment lower endpoint.
1670      IF I=J THEN Nextgroup       ! If the current segment is sorted
, then
1680      T$=A$(I+1)                  !  sort the next segment.
1690      IF Incdec=0 THEN D6
1700 I6:  IF A$(I)<=T$ THEN Inc
1710      GOTO 1730
1720 D6:  IF A$(I)>=T$ THEN Inc       ! Check to see if next element is
in order.
1730      K=I                         ! Insert element in otherwise sort
ed list.
```

```
1740 Copy: A$(K+1)=A$(K)              ! This section bumps the array up.
1750       K=K-1                      ! Prepare to bump next element.
1760       IF Incdec=0 THEN D7
1770 I7:   IF T$<A$(K) THEN Copy
1780       GOTO 1800
1790 D7:   IF T$>A$(K) THEN Copy      ! Check to see if insertion is h
ere.
1800       A$(K+1)=T$                 ! If so, then insert.
1810       GOTO Inc
1820 Nextgroup: M=M-1                 ! Pop the stack.
1830            IF M=0 THEN Out       ! Check for end conditions.
1840            I=L(M)                ! Restore the
1850            J=U(M)                !  previous endpoints.
1860            GOTO 1630
1870 Out: SUBEXIT
```

Method:

This subroutine uses a method of sorting which is similar to the QUICKERSORT algorithm by R.S. Scowen [2], which in turn is similar to an algorithm by Hibbard [3,4] and to Hoare's QUICKSORT [5]. The subroutine was translated from a FORTRAN listing of Algorithm 347 in the Communications of the ACM [1].

REFERENCES:


1. Singleton, Richard C. Certification of Algorithm 347. Comm. ACM Vol. 12, Number 3, March 1969, pp. 185-187.
2. Scowen, R.S. Algorithm 271, Quickersort. Comm. ACM 8 (Nov 1965), p. 669.
3. Hibbard, Thomas N. Some combinatorial properties of certain trees with applications to searching and sorting. Journal of ACM 9 (Jan 1962), 13-28.
4. Hibbard, Thomas N. An empirical study of minimal storage sorting. Comm. ACM 6 (May 1963), 206-213.
5. Hoare, C.A.R. Algorithms 63, Partition, and 64, Quicksort. Comm. ACM 4 (July 1961) 321.

Driver Utilization:

File Name: QSRTSD

This driver allows the user to either enter an array of his own, or have a random array generated. The array will then be printed, and the driver will call the subprogram Stringsort_q which will sort the array. Upon returning to the driver, the sorted array will be printed and titled.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "QSRTSD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c) Go to step 5.
5. When "Enter the lower subscript of the string vector" appears

in the display area:

a. Enter: The lower subscript of the vector.

b. Press: CONT

6. When "Enter the upper subscript of the string vector" appears in the display area:

   a. Enter: The upper subscript of the vector

   b. Press: CONT

7. When "Length of each string?" appears in the display area:

   a. Enter: The number of characters in each string.

   b. Press: CONT

8. When "Do you want to select your own string vector (Y/N)?" appears in the display area:

   a. If you want to enter your own string vector:

      1) Enter: Y

      2) Press: CONT

      3) Go to step 9.

      or

   a. If you do not want to enter your own vector, but would rather have the machine generate one:

      1) Enter: N

      2) Press: CONT

      3) Go to step 11.

9. When "ELEMENT # i?" appears in the display area:

   a. Enter: The ith element of the string vector.

   b. Press: CONT

   c. Repeat step 9 as often as necessary.

10. When "Enter the subscript?" appears in the display area:

   a. If you wish to make any changes in the vector:

      1) Enter: The subscript of the element you wish to change.

      2) Press: CONT

      3) Go to step 9.

      or

   a. If you do not wish to make any changes:

      1) Press: CONT

      2) Go to step 11.

11. When "Do you want to sort in ascending order or descending order (A/D)?" appears in the display area:

    a. If you want to sort in ascending order:

        1) Enter: A

        2) Press: CONT

        or

    a. If you want to sort in descending order:

        1) Enter: D

        2) Press: CONT

12. The sorted string vector will be printed and titled.

13. When "Do you want to sort the string vector again (Y/N)?" appears in the display:

    a. If you want to sort the string vector again:

        1) Type: Y

        2) Press: CONT

        3) Go to step 10.

        or

    a. If you do not want to sort the string vector again:

        1) Type: N

        2) Press: CONT

        3) The program stops.

The following string vector is sorted in ascending order:

```
UNSORTED STRING VECTOR
#     1      Friends
#     2      Romans
#     3      countrymen
#     4      lend
#     5      me
#     6      your
#     7      ears.
#     8      I
#     9      come
#    10      to
#    11      bury
#    12      Caesar
#    13      not
#    14      to
#    15      praise
#    16      him


SORTED STRING VECTOR
#     1      Caesar .
#     2      Friends
#     3      I
#     4      Romans
#     5      bury
#     6      come
#     7      countrymen
#     8      ears.
#     9      him
#    10      lend
#    11      me
#    12      not
#    13      praise
#    14      to
#    15      to
#    16      your
```

# Driver Listing

```
10 GET "QSORTS",1500
20 PRINTER IS 16
30 PRINT PAGE,"Any instructions or prompts that occur while this progr
am is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONT without entering"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90 S=16
100 INPUT "Printer select code?",S
110  S=INT(S)
120  IF (S>16) OR (S<0) THEN 90
130  IF S<>16 THEN 151
140  Top$="N"
150  GOTO 240
151  PRINTER IS S
152  PRINT
153  PRINTER IS 16
160  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
170  PRINT "has top-of-form generation and whether or not you wish to
make"
180  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
190  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
200  PRINT "generation, enter an N and press CONT."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 210
240  PRINT PAGE
250  PRINT "Below you are asked to enter the lower and upper subscript
s of the"
260  PRINT "one-dimensional string vector you wish to sort.  The subsc
ripts may be"
270 PRINT "negative, positive, or both.  The only restriction is that
the lower"
280  PRINT "subscript must be less than or equal to the upper subscrip
t",LIN(4)
290 INPUT "Enter the lower subscript of the string array",L
300 INPUT "Enter the upper subscript of the string array",U
```

```
310  IF U>=L THEN 360
320  BEEP
330  DISP "ILLEGAL RANGE"
340  WAIT 700
350  GOTO 290
360  PRINT PAGE
370  PRINT "Now you are asked to enter the length of the each member o
f"
380  PRINT "the string vector (in characters).  If you press CONT"
390  PRINT "without entering anything, the default string length of 18
"
400  PRINT "will be assumed.  Negative numbers or zero will not be acc
epted."
410  Length=18
420  INPUT "Length of each string?",Length
430  IF Length>0 THEN 480
440  BEEP
450  DISP "ZERO OR NEGATIVE STRING LENGTH NOT ALLOWED"
460  WAIT 700
470  GOTO 410
480  PRINT PAGE
490 CALL Sortdr(L,U,Length,S,Top$)
491  BEEP
492  DISP "PROGRAM COMPLETED"
500 END
510 SUB Sortdr(L,U,Length,S,Top$)
520 DIM A$(L:U)[Length],B$(L:U)[Length]
530  RANDOMIZE
540  PRINT "If you wish to enter your own string vector, enter Y in re
sponse to the"
550  PRINT "question below.  If you don't wish to enter your own strin
g vector, but"
560  PRINT "would rather have the machine generate a random string vec
tor, enter N."
570  PRINT "If you press CONT without entering a number, Y will be ass
umed"
580  PRINT "as the response.",LIN(4)
590  A$="Y"
600     INPUT "Do you want to select your own string vector (Y/N)?",A$
610     IF (A$="Y") OR (A$="y") THEN Manual
620     IF (A$="N") OR (A$="n") THEN Auto
630     GOTO 590
640 Manual: PRINT PAGE
650         PRINTER IS S
660         FOR I=L TO U
670            DISP "ELEMENT #";I;
680            INPUT A$(I)
```

```
690             PRINT USING 700;I,A$(I)
700             IMAGE "#",M4D,5X,K
710           NEXT I
720 PRINTER IS 16
730  IF S<>16 THEN PRINT PAGE
740 PRINT LIN(1),"If you want to make any changes, enter the subscript
 of the"
750 PRINT "element you wish to change.  Otherwise, press CONT without"
760  PRINT "entering a number.",LIN(2)
770  PRINTER IS S
780 Input: I=3.141592654
790         INPUT "Enter the subscript",I
800         IF I=3.141592654 THEN Run
810         IF (I>=L) AND (I<=U) THEN 860
820         BEEP
830         DISP "ILLEGAL SUBSCRIPT"
840         WAIT 700
850         GOTO Input
860         DISP "ELEMENT #";I;
870         INPUT A$(I)
880         PRINT USING 700;I,A$(I)
890         GOTO Input
900 Auto: FOR I=L TO U
910           FOR J=1 TO Length
920           X=INT(128*RND)
930           IF (X<65) OR (X>90) AND (X<97) OR (X>122) THEN 920
940           A$(I)[J,J]=CHR$(X)
950           NEXT J
960        NEXT I
970 Run: PRINTER IS S
980       IF (S<>16) AND ((Topy$="y") OR (Top$="Y")) THEN PRINT PAGE;
990       PRINT LIN(2),"UNSORTED STRING VECTOR"
1000      CALL Output(A$(*),L,U)
1010      PRINTER IS 16
1020      IF S<>16 THEN PRINT PAGE;
1030      IF S=16 THEN PRINT LIN(2);
1040      PRINT "If you want the vector to be sorted in ascending order
,"
1050      PRINT "enter A.  If you want the vector to be sorted in desce
nding"
1060      PRINT "order, enter D.  If you press CONT without entering"
1070      PRINT "anything, A will be the assumed response.",LIN(2)
1080      A$="A"
1090      INPUT "Do you want to sort in ascending or descending order (
A/D)?",A$
1100      IF (UPC$(A$)="D") OR (UPC$(A$)="A") THEN 1150
1110      BEEP
```

```
1120      DISP "ILLEGAL RESPONSE"
1130      WAIT 700
1140      GOTO 1080
1150      IF S<>16 THEN PRINT PAGE
1160      PRINTER IS S
1170      Incdec=1
1180      IF (A$="D") OR (A$="d") THEN Incdec=0
1190      FOR I=L TO U
1200      B$(I)=A$(I)
1210      NEXT I
1220      DISP "SORTING"
1230      CALL Stringsort_q(A$(*),L,U,Length,Incdec)
1240      IF (S<>16) AND ((Topy$="y") OR (Top$="Y")) THEN PRINT PAGE;
1250      PRINT LIN(2),"SORTED STRING VECTOR"
1260      CALL Output(A$(*),L,U)
1270      IF (S<>16) AND ((Topy$="y") OR (Top$="Y")) THEN PRINT PAGE
1280      INPUT "Do you want to sort the string vector again (Y/N)?",A$
1290      IF UPC$(A$)="N" THEN 1410
1300      IF UPC$(A$)="Y" THEN 1350
1310      BEEP
1320      DISP "ILLEGAL RESPONSE"
1330      WAIT 700
1340      GOTO 1280
1350      FOR I=L TO U
1360      A$(I)=B$(I)
1370      NEXT I
1380      PRINT LIN(2),"ORIGINAL STRING VECTOR"
1390      CALL Output(A$(*),L,U)
1400      GOTO 720
1410      PRINTER IS 16
1420 SUBEXIT
1430 SUB Output(A$(*),L,U)
1440   FOR I=L TO U
1450        PRINT USING 1460;I,A$(I)
1460        IMAGE "#",M4D,5X,K
1470   NEXT I
1480 SUBEXIT
1490 SUBEND
```

# MINIMAL STORAGE SORT OF A STRING VECTOR

Subprogram Name: Stringsort_s

This subprogram allows the user to sort a one-dimensional string vector (or any subset of that vector) into either ascending or descending order.

This sort is slower than the Fast Sort on page 87 but it takes much less memory. The subprogram itself takes 2788 bytes less memory, and no auxiliary storage is needed, as in the Fast Sort.

Subprogram Utilization:

File Name: SSORTS

Calling Syntax: CALL Stringsort_s (A$(*),A,B,C,D)

Input Parameters:

A$(*) - The first parameter in the parameter list must be a one-dimensional string array. This is the string vector which will be sorted.

A - The second parameter must be a full-precision variable, constant, or expression. A represents the lower bound of the sort.

B - The third parameter must be a full-precision variable, constant, or expression. B represents the upper bound of the sort.

C - The fourth parameter must be a full-precision variable, constant, or expression. C represents the length of each member of the string vector.

D - The last parameter must be a full-precision variable, constant, or expression. D indicates whether the sort will be ascending or

descending. If D = 0, the sort will be descending. If D = 1, the sort will be ascending. If D is anything else, the vector will not be sorted.

Output Parameters:

A$(*) - After this subprogram is executed, this string vector will be sorted from elements A through B (if the value passed in as D is Ø or 1 (see the section under Input Parameters)).

Local Variables:

H - $2^{(S-1)}$ The increment of the diminishing increment sort
I - Temporary subscript
J - Counter on sort loop
K$ - Temporary variable (for switching)
S - Counter on diminishing increment loop
T - $Log_2$ of the number of elements to be sorted.

Special Considerations and Programming Hints:

1. If the lower bound of the sort is greater than or equal to the upper bound of the sort, the returned array will be unchanged from its input state. If the fourth parameter passed in through the subprogram call (the string length) is less than 1, then an error will occur since a string can not be dimensioned to have a zero or negative length.
2. As mentioned in the introduction, this sort is slower than the Fast Sort of a string vector on page 87, but it also takes much less memory. If memory is not a concern, the Fast Sort is recommended. (Note: See GRAPH 1 immediately following this section for timing comparisons.)
3. This sort may be made faster by removing the capability of sorting both in increasing and decreasing order. An extra listing is included showing which lines need to be deleted in order to accomplish this. The modified listing (which may be found immediately following GRAPH 2) will sort in ascending order. To sort in descending order, merely re-

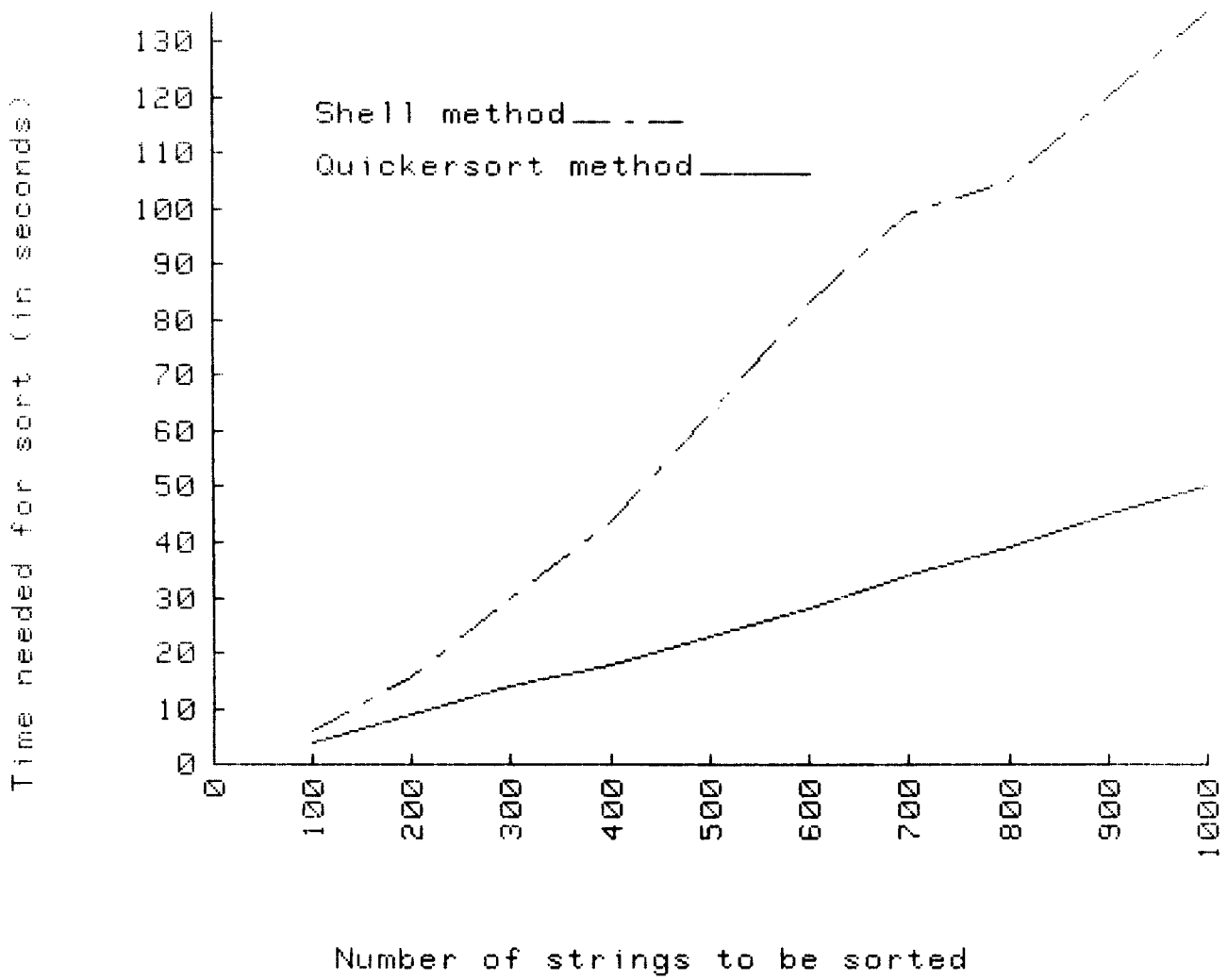
verse the inequality comparison in statement Increase to read:

```
Increase:  IF  K$< = A$(I)  THEN Insert
```

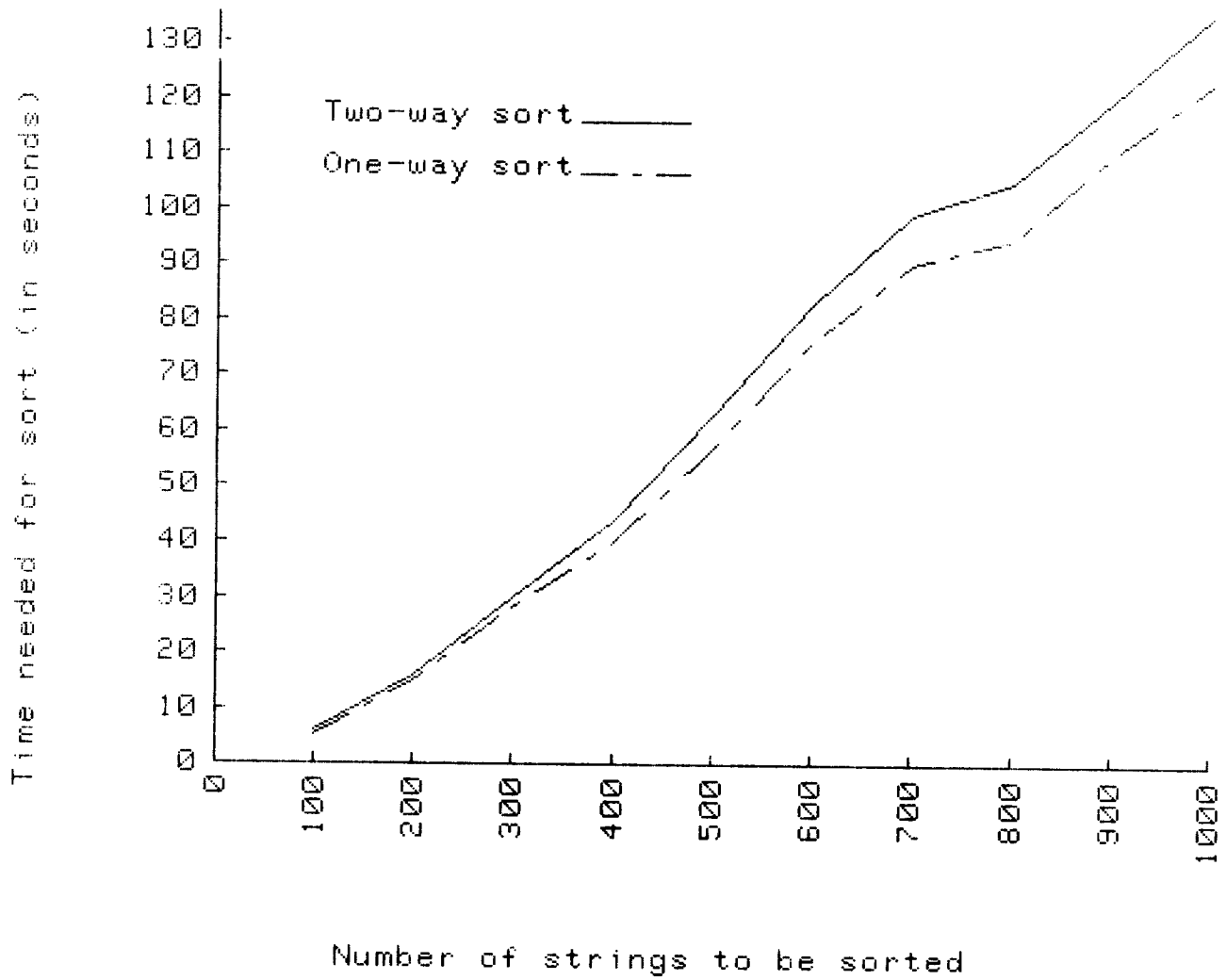
GRAPH 2 immediately following this section shows how much improvement in speed is gained by using the procedure outlined above:

4. System Configuration:
   Standard Memory option
   (Optional) Printer

GRAPH 1 - Quickersort vs. Shell sort



GRAPH 1 - Two-way sort vs. One way sort



## Modified Subprogram Listing (Ascending sort only)

This listing shows those lines which can be removed to make the sort faster. (The deleted lines would allow the user to sort both in ascending and descending order if they were left in.)

```
1010 SUB Stringsort_s(A$(*),I1,J1,Length,Incdec)
1020 REM  This subroutine sorts a one-dimensional vector (or any subse
t of that
1030 REM  vector) into either ascending or descending order.  Shell's
sort
1040 REM  (diminishing increment sort) is used.
1050 DIM K$[Length]
1060 IF (Incdec<>0) AND (Incdec<>1) OR (J1<=I1) THEN SUBEXIT
1070 T=INT(LOG(J1+1-I1)/LOG(2))  !  Compute the number of diminishing
increments
1080                             !    to be used in the sort.
1090 FOR S=T TO 1 STEP -1
1100    H=2^(S-1)                !  The set of increments to be used i
s ...,16,8
1110                             !    4,2,1
1120    FOR J=H+I1 TO J1
1130       I=J-H
1140       K$=A$(J)
1150 Decide:IF Incdec=0 THEN Decrease
1160 Increase:IF K$>=A$(I) THEN Insert
1170          GOTO Switch
1180 Decrease:IF K$<=A$(I) THEN Insert
1190 Switch:  A$(I+H)=A$(I)
1200          I=I-H
1210          IF I>=I1 THEN Decide
1220 Insert:  A$(I+H)=K$
1230    NEXT J
1240 NEXT S
1250 SUBEXIT
```

# Annotated Listing of Subprogram Stringsort_ s

```
1010 SUB Stringsort_s(A$(*),I1,J1,Length,Incdec)
1020 REM  This subroutine sorts a one-dimensional vector (or any subse
t of that
1030 REM  vector) into either ascending or descending order.  Shell's
sort
1040 REM  (diminishing increment sort) is used.
1050 DIM K$[Length]
1060 IF (Incdec<>0) AND (Incdec<>1) OR (J1<=I1) THEN SUBEXIT
1070 T=INT(LOG(J1+1-I1)/LOG(2))  !  Compute the number of diminishing
increments
1080                             !    to be used in the sort.
1090 FOR S=T TO 1 STEP -1
1100    H=2^(S-1)                !  The set of increments to be used i
s ...,16,8
1110                             !    4,2,1
1120    FOR J=H+I1 TO J1
1130       I=J-H
1140       K$=A$(J)
1150 Decide:IF Incdec=0 THEN Decrease
1160 Increase:IF K$>=A$(I) THEN Insert
1170          GOTO Switch
1180 Decrease:IF K$<=A$(I) THEN Insert
1190 Switch:  A$(I+H)=A$(I)
1200          I=I-H
1210          IF I>=I1 THEN Decide
1220 Insert:  A$(I+H)=K$
1230    NEXT J
1240 NEXT S
1250 SUBEXIT
```

Method:

The method used in this subprogram is Donald Shell's diminishing increment sort. First, a sequence of diminishing increments $h_t$, $h_{t-1}$, ..., $h_1$ is selected. (In this case, t is $\log_2$ N, where N is the number of elements to be sorted, and the h's are $2^{t-1}$, $2^{t-2}$, ....$2^0$.) Then, for each h, a straight insertion sort is performed on those elements of the array which are h elements apart. By the time the last h has been used, the array is sorted. (Note: the other h's may be anything, as long as they are diminishing, but $h_1$ must <u>always</u> be 1.)

REFERENCE:

Knuth, Donald E., THE ART OF COMPUTER PROGRAMMING Vol. 3 (SORTING AND SEARCHING) (Addison-Wesley 1973) pp. 84-85.

Driver Utilization:

File Name: SSRTSD

This driver allows the user to either enter a string array of his own, or have a random array generated. The array will then be printed, and the driver will call the subprogram Stringsort_s which will sort the string array. Upon returning to the driver, the sorted array will be printed and titled.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SSRTSD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display are:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.

         or
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c) Go to step 5.

5. When "Enter the lower subscript of the string vector" appears in the display area:
   a. Enter: The lower subscript of the vector.
   b. Press: CONT
6. When "Enter the upper subscript of the string vector" appears in the display area:
   a. Enter: The upper subscript of the vector.
   b. Press: CONT
7. When "Length of each string?" appears in the display area:
   a. Enter: The number of characters in each string.
   b. Press: CONT
8. When "Do you want to select your own string vector (Y/N)?" appears in the display area:
   a. If you want to enter your own string vector:
      1) Enter: Y
      2) Press: CONT
      3) Go to step 9.

      or

   a. If you do not want to enter your own vector, but would rather have the machine generate one:
      1) Enter: N
      2) Press: CONT
      3) Go to step 11.
9. When "ELEMENT # i?" appears in the display area:
   a. Enter: The ith element of the string vector.
   b. Press: CONT
   c. Repeat step 9 as often as necessary.
10. When "Enter the subscript?" appears in the display area:
   a. If you wish to make any changes in the vector:
      1) Enter: The subscript of the element you wish to change.
      2) Press: CONT
      3) Go to step 9.

      or

   a. If you do not wish to make any changes:
      1) Press: CONT
      2) Go to step 11.

11. When "Do you want to sort in ascending order or descending order (A/D)?" appears in the display area:
    a. If you want to sort in ascending order:
        1) Enter: A
        2) Press: CONT

        or

    a. If you want to sort in descending order:
        1) Enter: D
        2) Press: CONT
12. The sorted string vector will be printed and titled.
13. When "Do you want to sort the string vector again (Y/N)?" appears in the display:
    a. If you want to sort the string vector again:
        1) Type: Y
        2) Press: CONT
        3) Go to step 10.

        or

    a. If you do not want to sort the string vector again:
        1) Type: N
        2) Press: CONT
        3) The program stops.

## EXAMPLE

The following string array is sorted in descending order.

```
UNSORTED STRING VECTOR
#     1      The
#     2      brave
#     3      boy
#     4      seized
#     5      the
#     6      browny
#     7      Indian
#     8      and
#     9      hurled
#    10      him
#    11      over
#    12      the
#    13      beetling
#    14      precipice
#    15      into
#    16      the
#    17      seething
#    18      depths
#    19      below

SORTED STRING VECTOR
#     1      the
#     2      the
#     3      the
#     4      seized
#     5      seething
#     6      precipice
#     7      over
#     8      into
#     9      hurled
#    10      him
#    11      depths
#    12      browny
#    13      brave
#    14      boy
#    15      below
#    16      beetling
#    17      and
#    18      The
#    19      Indian
```

# Driver Listing

```
10 GET "SSORTS",1500
20 PRINTER IS 16
30 PRINT PAGE,"Any instructions or prompts that occur while this progr
am is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONT without entering"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90 S=16
100 INPUT "Printer select code?",S
110  S=INT(S)
120  IF (S>16) OR (S<0) THEN 90
130  IF S<>16 THEN 151
140  Top$="N"
150  GOTO 240
151  PRINTER IS S
152  PRINT
153  PRINTER IS 16
160  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
170  PRINT "has top-of-form generation and whether or not you wish to
make"
180  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
190  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
200  PRINT "generation, enter an N and press CONT."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 210
240  PRINT PAGE
250  PRINT "Below you are asked to enter the lower and upper subscript
s of the"
260  PRINT "one-dimensional string vector you wish to sort.  The subsc
ripts may be"
270 PRINT "negative, positive, or both.  The only restriction is that
the lower"
280  PRINT "subscript must be less than or equal to the upper subscrip
t",LIN(4)
290 INPUT "Enter the lower subscript of the string array",L
300 INPUT "Enter the upper subscript of the string array",U
```

```
310   IF U>=L THEN 360
320   BEEP
330   DISP "ILLEGAL RANGE"
340   WAIT 700
350   GOTO 290
360   PRINT PAGE
370   PRINT "Now you are asked to enter the length of the each member o
f"
380   PRINT "the string vector (in characters).  If you press CONT"
390   PRINT "without entering anything, the default string length of 18
"
400   PRINT "will be assumed.  Negative numbers or zero will not be acc
epted."
410   Length=18
420   INPUT "Length of each string?",Length
430   IF Length>0 THEN 480
440   BEEP
450   DISP "ZERO OR NEGATIVE STRING LENGTH NOT ALLOWED"
460   WAIT 700
470   GOTO 410
480   PRINT PAGE
490 CALL Sortdr(L,U,Length,S,Top$)
491   BEEP
492   DISP "PROGRAM COMPLETED"
500 END
510 SUB Sortdr(L,U,Length,S,Top$)
520 DIM A$(L:U)[Length],B$(L:U)[Length]
530   RANDOMIZE
540   PRINT "If you wish to enter your own string vector, enter Y in re
sponse to the"
550   PRINT "question below.  If you don't wish to enter your own strin
g vector, but"
560   PRINT "would rather have the machine generate a random string vec
tor, enter N."
570   PRINT "If you press CONT without entering a number, Y will be ass
umed"
580   PRINT "as the response.",LIN(4)
590   A$="Y"
600      INPUT "Do you want to select your own string vector (Y/N)?",A$
610      IF (A$="Y") OR (A$="y") THEN Manual
620      IF (A$="N") OR (A$="n") THEN Auto
630      GOTO 590
640 Manual: PRINT PAGE
650          PRINTER IS S
660          FOR I=L TO U
670             DISP "ELEMENT #";I;
680             INPUT A$(I)
```

```
690             PRINT USING 700;I,A$(I)
700             IMAGE "#",M4D,5X,K
710           NEXT I
720 PRINTER IS 16
730  IF S<>16 THEN PRINT PAGE
740 PRINT LIN(1),"If you want to make any changes, enter the subscript
 of the"
750 PRINT "element you wish to change.  Otherwise, press CONT without"
760  PRINT "entering a number.",LIN(2)
770  PRINTER IS S
780 Input: I=3.141592654
790         INPUT "Enter the subscript",I
800         IF I=3.141592654 THEN Run
810         IF (I>=L) AND (I<=U) THEN 860
820         BEEP
830         DISP "ILLEGAL SUBSCRIPT"
840         WAIT 700
850         GOTO Input
860         DISP "ELEMENT #";I;
870         INPUT A$(I)
880         PRINT USING 700;I,A$(I)
890         GOTO Input
900 Auto: FOR I=L TO U
910           FOR J=1 TO Length
920           X=INT(128*RND)
930           IF (X<65) OR (X>90) AND (X<97) OR (X>122) THEN 920
940           A$(I)[J,J]=CHR$(X)
950           NEXT J
960        NEXT I
970 Run: PRINTER IS S
980       IF (S<>16) AND ((Topy$="y") OR (Top$="Y")) THEN PRINT PAGE;
990       PRINT LIN(2),"UNSORTED STRING VECTOR"
1000      CALL Output(A$(*),L,U)
1010      PRINTER IS 16
1020      IF S<>16 THEN PRINT PAGE;
1030      IF S=16 THEN PRINT LIN(2);
1040      PRINT "If you want the vector to be sorted in ascending order
,"
1050      PRINT "enter A.  If you want the vector to be sorted in desce
nding"
1060      PRINT "order, enter D.  If you press CONT without entering"
1070      PRINT "anything, A will be the assumed response.",LIN(2)
1080      A$="A"
1090      INPUT "Do you want to sort in ascending or descending order (
A/D)?",A$
1100      IF (UPC$(A$)="D") OR (UPC$(A$)="A") THEN 1150
1110      BEEP
```

```
1120      DISP "ILLEGAL RESPONSE"
1130      WAIT 700
1140      GOTO 1080
1150      IF S<>16 THEN PRINT PAGE
1160      PRINTER IS S
1170      Incdec=1
1180      IF (A$="D") OR (A$="d") THEN Incdec=0
1190      FOR I=L TO U
1200      B$(I)=A$(I)
1210      NEXT I
1220      DISP "SORTING"
1230      CALL Stringsort_s(A$(*),L,U,Length,Incdec)
1240      IF (S<>16) AND ((Topy$="y") OR (Top$="Y")) THEN PRINT PAGE;
1250      PRINT LIN(2),"SORTED STRING VECTOR"
1260      CALL Output(A$(*),L,U)
1270      IF (S<>16) AND ((Topy$="y") OR (Top$="Y")) THEN PRINT PAGE
1280      INPUT "Do you want to sort the string vector again (Y/N)?",A$
1290      IF UPC$(A$)="N" THEN 1410
1300      IF UPC$(A$)="Y" THEN 1350
1310      BEEP
1320      DISP "ILLEGAL RESPONSE"
1330      WAIT 700
1340      GOTO 1280
1350      FOR I=L TO U
1360      A$(I)=B$(I)
1370      NEXT I
1380      PRINT LIN(2),"ORIGINAL STRING VECTOR"
1390      CALL Output(A$(*),L,U)
1400      GOTO 720
1410      PRINTER IS 16
1420 SUBEXIT
1430 SUB Output(A$(*),L,U)
1440    FOR I=L TO U
1450        PRINT USING 1460;I,A$(I)
1460        IMAGE "#",M4D,5X,K
1470    NEXT I
1480 SUBEXIT
1490 SUBEND
```

HEWLETT hp PACKARD

# POLYNOMIAL ROOTFINDER: SILJAK'S METHOD

Subprogram Name: Siljak

This subprogram will find all roots, Z, of polynomials of the form.

$$a_0 + i\,b_0 + (a_1 + ib_1)\,Z + (a_2 + ib_2)\,Z^2 + \ldots + (a_n + ib_n)\,Z^n = \emptyset .$$

Roots are found by expressing the polynomials in terms of Siljak functions and using the method of steepest descent to determine the zeros.

The user is required to enter the complex coefficients of the polynomial as well as its degree. Tolerances for the root and for function evaluations and the maximum number of iterations are also needed.

Subprogram Utilization:

File Name: Siljak

Calling Syntax: CALL Siljak (N, Rcoef(*), Icoef(*), Tola, Tolf, Itmax, Rroot(*), Iroot(*)).

Input Parameters:

N - Degree of polynomial; number of roots to be found.

Rcoef(*) - Vector containing the real parts of the coefficients of the polynomial where the subscript corresponds to the exponent of the variable; subscripted from 0 to N. (e.g., For $a_0 + ib_0 + (a_1 + ib_1)\,Z + (a_2 + ib_2)\,Z^2 + \ldots + (a_n + ib_n)\,Z^n = 0$, Rcoef (3) would be the real part of the coefficient of the $Z^3$ term, $a_3$.)

Icoef(*) - Vector containing the imaginary parts of the coefficients of the polynomial where the subscript corresponds to the exponent of the variable; subscripted from $\emptyset$ to N.

Tola - Tolerance for the root.

Tolf - Tolerance for the function evaluation.

Itmax - Maximum number of iterations permitted in searching for any one root.

Output Parameters:

Rroot(*) - Vector containing the real parts of the roots of the polynomial, subscripted from 1 to N.

Iroot(*) - Vector containing the imaginary parts of the roots of the polynomial; subscripted from 1 to N.

Local Variables:

A - Temporary storage for Rcoef (K), K=Ø, N-1

B - Temporary storage for Icoef (K), K=Ø, N-1

Baddta - Used in BAD DATA CHECK.
Baddta = 1 implies bad data.
Baddta = Ø implies input data reasonable

C - Temporary storage for Icoef (K), K=1 to N

D - Temporary storage for Rcoef (K), K=1 to N

Deltax - $\Delta x$

Deltay - $\Delta y$

F - $u^2 + v^2$, function value for Siljak Functions

G - Former value of F; used to test for convergence

I,K - Loop counters

L - Counter to determine if after Itmax iterations convergence has not occurred.

M - Counter to determine if 2Ø successive quarterings of the search increment have not reduced the function value.

P - $\frac{\partial u}{\partial x}$

Q - $\frac{\partial V}{\partial x}$

T - Temporary storage

U - Real part of F; temporary storage for Rcoef (K+1), K=Ø, N-1

V - Imaginary part of F; temporary storage for Icoef (K+1), K=Ø, N-1

X
Y - x+iy is the current root approximation

Z - $x^2 + y^2$ used in computation of Siljak Functions:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial x}\right)^2$$

Xsiljak(*)
Ysiljak(*) - Siljak Function where $Z^k$ = Xsiljak(K) + i Ysiljak(K)

Special Considerations:

1. Upon entry into subprogram Siljak, there is a bad data check. If the check detects any "nonsense" data, the following error message will be printed and the program pauses:

   "ERROR IN SUBPROGRAM Siljak"

   N = , Tola = , Tolf = , Iter =

   The data may be corrected from the keyboard (e.g. TOLA=1 E-6, EXECUTE). When CONT is pressed, the program will resume execution at the next line.

2. The subprogram has been set to quarter the interval size at most 2Ø times. If this maximum is exceeded, an error message is printed:

   "ERROR IN SUBPROGRAM Siljak."
   "THE INTERVAL SIZE HAS BEEN QUARTERED 2Ø TIMES AND
   THE TOLERANCE FOR FUNCTIONAL EVALUATIONS IS STILL
   NOT MET."
   Tolf = , U = , V =

The subprogram is then exited with all known information. All roots not found will contain the value 9.999999 E 99.

If more than 20 quarterings are required, there is probably some unusual behavior in the function.

3. If the maximum number of iterations has been exceeded, the following error message will be printed and Siljak will be exited with all known information.

```
"ERROR IN SUBPROGRAM Siljak."
"MAXIMUM # OF ITERATIONS HAS BEEN EXCEEDED."
L =                          Itmax =
```

Again, all roots not located will contain the value 9.999999 E 99.

There are two possible solutions here. First, increase Itmax, the maximum number of iterations allowed; or second, allow larger error tolerances, Tola and Tolf. The iteration counter, L, is reset after each root is located, so attempting to solve a polynomial of large degree should not cause this error.

4. System Configuration:
   Standard Memory option
   (Optional) Printer

# Annotated Listing of Subprogram Siljak

```
10   SUB Siljak(N,Rcoef(*),Icoef(*),Tola,Tolf,Itmax,Rroot(*),Iroot(*))
20   DIM Xsiljak(0:N),Ysiljak(0:N)
30   ! ************************************************
40   ! *** POLYNOMIAL ROOTFINDER USING SILJAK'S METHOD.
50   ! ************************************************
60   !
70   !
80   ! *** BAD DATA CHECK.
90   Baddta=(N<=0) OR (Tola<=0) OR (Tolf<=0) OR (Itmax<=0)
100  IF Baddta=0 THEN 230
110  ! *** PRINT ERROR MESSAGE AND PAUSE.
120  ! *** USE MAY CORRECT DATA AND CONTINUE.
130  PRINT LIN(2),"ERROR IN SUBPROGRAM Siljak."
140  PRINT "N=";N,"Tola=";Tola
150  PRINT "Tolf=";Tolf,"Itmax=";Itmax,LIN(2)
160  PAUSE
161  GOTO 90
170  !
180  !
190  ! *** BEGIN SUBPROGRAM.
200  ! *** INITIALIZE LOCAL VARIABLES.
210  ! *** Rroot(*) AND Iroot(*) HOLD THE ROOT APPROXIMATIONS.
220  ! *** INITIALIZE Rroot(*) AND Iroot(*) TO 9.999999E99.
230  MAT Rroot=(9.999999E99)
240  MAT Iroot=(9.999999E99)
250  !
260  !
270  Nn=N
280  ! *** SPECIAL CHECK FOR N=1.
290  IF N=1 THEN 1400
300  !
310  !
320  ! *** Xsiljak(*) AND Ysiljak(*) HOLD THE SILJAK COEFFICIENTS.
330  Y=Ysiljak(1)=Xsiljak(0)=1
340  X=Xsiljak(1)=.1
350  Ysiljak(0)=L=0
360  !
370  !
380  ! *** BRANCH TO COMPUTATION OF SILJAK COEFFICIENTS.
390  GOSUB Siljak
400  ! *** G GETS FORMER VALUE OF F, INITIALIZE AND INCREMENT
410  ! *** ITERATION COUNTER.
420  G=F
430  M=0=P=0
440  L=L+1
450  ! *** Z=(DU/DX)^2+(DV/DX)^2.
460  FOR K=1 TO N
```

```
470       P=P+K*(Rcoef(K)*Xsiljak(K-1)-Icoef(K)*Ysiljak(K-1))
480       Q=Q+K*(Rcoef(K)*Ysiljak(K-1)+Icoef(K)*Xsiljak(K-1))
490    NEXT K
500    Z=P*P+Q*Q
510    ! *** Deltax AND Deltay ARE THE RESPECTIVE CHANGES IN X AND CHANG
ES IN Y.
520    Deltax=-(U*P+V*Q)/Z
530    Deltay=(U*Q-V*P)/Z
540    ! *** INCREMENT SUCCESSIVE QUARTERING COUNTER.
550    M=M+1
560    !
570    !
580    ! *** NEW ROOT APPROXIMATIONS X,Y LOADED INTO Xsiljak(1) AND Ysil
jak(1).
590    Xsiljak(1)=X+Deltax
600    Ysiljak(1)=Y+Deltay
610    ! *** RECOMPUTE SILJAK COEFFICIENTS.
620    GOSUB Siljak
630    !
640    !
650    ! *** IF NEW ERROR ESTIMATE GREATER THAN OLD, QUARTER THE SIZE
660    ! *** OF Deltax AND Deltay.
670    IF F>=G THEN 870
680    ! *** CHECK IF INCREMENTS ARE SMALL ENOUGH TO SATISFY THE
690    ! *** STOPPING CONDITIONS.
700    IF (ABS(Deltax)<Tola) AND (ABS(Deltay)<Tola) THEN 1130
710    ! *** CHECK IF MAXIMUM NUMBER OF ITERATIONS HAS BEEN EXCEEDED.
720    ! *** IF SO, PRINT ERROR MESSAGE AND PAUSE.
730    IF L>Itmax THEN 1040
740    !
750    !
760    ! *** ITERATE AGAIN.
770    X=Xsiljak(1)
780    Y=Ysiljak(1)
790    GOTO 420
800    !
810    !
820    ! *** QUARTER THE INTERVAL SIZE AND TRY AGAIN.
830    ! *** MAXIMUM # OF QUARTERINGS HAS BEEN SET AT 20.
840    ! *** IF THIS IS EXCEEDED, THEN A FINAL TEST IS MADE ON U AND V T
O
850    ! *** SEE IF THE FUNCTIONAL TOLERANCE IS SATISFIED.  IF NOT, AN
860    ! *** ERROR MESSAGE IS PRINTED AND THE PROGRAM EXITS.
870    IF M>20 THEN 910
880    Deltax=Deltax/4
890    Deltay=Deltay/4
900    GOTO 550
```

```
910  IF (ABS(U)<=Tolf) AND (ABS(V)<=Tolf) THEN 1130
920  !
930  !
940  ! *** PRINT ERROR MESSAGE AND PAUSE.
950  PRINT LIN(2),"ERROR IN SUBPROGRAM Siljak."
960  PRINT "THE INTERVAL SIZE HAS BEEN QUARTERED 20 TIMES AND "
970  PRINT "THE TOLERANCE FOR FUNCTIONAL EVALUATIONS IS STILL NOT MET.
"
980  PRINT "Tolf=";Tolf,"U=";U,"V=";V,LIN(2)
990  PAUSE
1000 !
1010 !
1020 ! *** MAXIMUM # OF ITERATIONS HAS BEEN EXCEEDED.
1030 ! *** PRINT ERROR MESSAGE AND PAUSE.
1040 PRINT LIN(2),"ERROR IN SUBROUTINE Siljak."
1050 PRINT "MAXIMUM # OF ITERATIONS HAS BEEN EXCEEDED."
1060 PRINT "L=";L,"Itmax=";Itmax,LIN(2)
1070 PAUSE
1080 GOTO 730
1090 !
1100 !
1110 ! *** ROOT FOUND.  STORE COMPUTED ROOT IN Rroot(*) AND Iroot(*)
1120 ! *** ARRAY ELEMENT.
1130 Rroot(N)=Xsiljak(1)
1140 Iroot(N)=Ysiljak(1)
1150 !
1160 !
1170 ! *** INTIALIZE VARIABLES FOR SYNTHETIC DIVISION.
1180 A=Rcoef(N)
1190 B=Icoef(N)
1200 Rcoef(N)=Icoef(N)=0
1210 X=Xsiljak(1)
1220 Y=Ysiljak(1)
1230 ! *** SYNTHETIC DIVISION TO CALCULATE NEW COEFFICENTS Rcoef(*)
1240 ! *** AND Icoef(*).
1250 FOR K=N-1 TO 0 STEP -1
1260    C=Rcoef(K)
1270    D=Icoef(K)
1280    U=Rcoef(K+1)
1290    V=Icoef(K+1)
1300    Rcoef(K)=A+X*U-Y*V
1310    Icoef(K)=B+X*V+Y*U
1320    A=C
1330    B=D
1340 NEXT K
1350 N=N-1
1360 ! *** REDUCE NUMBER OF COEFFICIENTS AND BEGIN AGAIN.
```

```
1370 IF N<>1 THEN 390
1380 ! *** SINCE DEGREE OF RESULTANT POLYNOMIAL IS ONE,
1390 ! *** COMPUTE FINAL ROOT ALGEBRAICALLY.
1400 A=Rcoef(0)
1410 U=Rcoef(1)
1420 B=Icoef(0)
1430 V=Icoef(1)
1440 T=U*U+V*V
1450 Rroot(1)=-(A*U+B*V)/T
1460 Iroot(1)=(A*V-U*B)/T
1470 N=Nn
1480 SUBEXIT
1490 !
1500 !
1510 ! *** SUBROUTINE TO COMPUTE SILJAK COEFFICIENTS.
1520 Siljak: Z=Xsiljak(1)*Xsiljak(1)+Ysiljak(1)*Ysiljak(1)
1530 T=2*Xsiljak(1)
1540 FOR K=0 TO N-2
1550     Xsiljak(K+2)=T*Xsiljak(K+1)-Z*Xsiljak(K)
1560     Ysiljak(K+2)=T*Ysiljak(K+1)-Z*Ysiljak(K)
1570 NEXT K
1580 U=V=0
1590 FOR K=0 TO N
1600     U=U+Rcoef(K)*Xsiljak(K)-Icoef(K)*Ysiljak(K)
1610     V=V+Rcoef(K)*Ysiljak(K)+Icoef(K)*Xsiljak(K)
1620 NEXT K
1630 F=U*U+V*V
1640 RETURN
1650 SUBEND
```

Method(s) and Formulae:

This subprogram will find all the roots Z of polynomials of the form $a_0 + ib_0 + (a_1 + ib_1) Z + (a_2 + ib_2) Z^2 + \ldots +$

$$(a_n + ib_n)Z^n = \emptyset.$$

Roots are found by expressing the polynomial in terms of Siljak functions and using the method of steepest descent to determine the zeros. Once a root is found, the polynomial is reduced by synthetic division and the process is repeated. The last root is computed algebraically. The algorithm is very accurate and stable; it will virtually always find the roots and the user is not required to provide an initial value. Multiple roots are found at some slightly reduced accuracy, and higher order polynomials may show some loss of accuracy as more roots are found. In general, the program will find "normally" spaced roots accurate to better than 6 decimal places. Newton's method could find the roots faster but convergence is not guaranteed and with Siljak's method, no a priori information such as the derivative, is necessary.

$$F(z) = \sum_{k=\emptyset}^{n} (a_k + ib_k) Z^k = \emptyset$$

Siljak Functions $x_k$ and $y_k$ are defined by $Z^k = X_k + iY_k$ and may be calculated recursively

$$x_0 = 1, \quad x_1 = .1, \quad y_0 = \emptyset, \quad y_1 = 1$$

$$X_{k+2} = 2x\, X_{k+1} - (x^2 + y^2) X_k$$

$$Y_{k+2} = 2x\, Y_{k+1} - (x^2 + y^2) Y_k$$

where x+iy are the root approximations

$$u = \sum_{k=\emptyset}^{n} (a_k X_k - b_k Y_k)$$

$$v = \sum_{k=\emptyset}^{n} (a_k Y_k + b_k X_k)$$

$$\frac{\partial u}{\partial x} = \sum_{k=1}^{n} k\,(a_k X_{k-1} - b_k Y_{k-1})$$

$$\frac{\partial v}{\partial x} = \sum_{k=1}^{n} k\,(a_k Y_{k-1} + b_k X_{k-1})$$

REFERENCES:

1. Moore, J.B., "A Convergent Algorithm for Solving Polynomial Equations", Journal of the Association for Computing Machinery. vol. 14, No. 2 (April, 1967), pp. 311-315.

Driver Utilization:

File Name: SILJAK

The driver "SILJAK" sets up the necessary input parameters for the subprogram Siljak as well as prints out the resulting outputs.

With each question, the user is only required to input the appropriate response.

User Instructions:

1. Insert the Utility Routines cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SILJAK: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "HAS Siljak ALREADY BEEN LINKED ON (Y/N)?" is displayed:
   a. Enter Y if the subprogram has already been linked on.
   b. Press: CONT
   c. Go to 5.

   or

   a. Enter N if the subprogram has not yet been linked on.
   b. Press: CONT
   c. At this point, subprogram Siljak contained in file "Siljak" will be linked on after the driver.
   d. Go to 5.
5. When "DEGREE OF POLYNOMIAL?" is displayed:
   a. Enter the degree of the polynomial; this is also the number of roots to be found.
   b. Press: CONT
6. When "MAX # OF ITERATIONS?" is displayed:
   a. Enter the maximum number of iterations allowed per root.
   b. Press: CONT

7. When "TOLERANCE FOR ROOTS?" is displayed:
   a. Enter the tolerance desired for the roots; a root is accepted if the difference in value of the root approximations of two successive iterations is less than this tolerance.
   b. Press: CONT
8. When "TOLERANCE FOR FUNCTIONAL EVALUATIONS?" is displayed:
   a. Enter the tolerance for the functional evaluations; a root x is accepted if $|F(x)| \leq$ this tolerance.
   b. Press: CONT
9. When "Rcoef( I)=?" (For I=1 to Degree of Polynomial) is displayed:
   a. Enter the appropriate real part of the coefficient; each subscript corresponds to the exponent of the variable; i.e., $(a_0 + b_0 i) + (a_1 + b_1 i) Z^1 + (a_2 + b_2 i) Z^2 + \ldots + (a_n + b_n i) Z^n$.
   b. Press: CONT
10. When "Icoef(I)=?" (For I=1 to Degree of Polynomial) is displayed:
   a. Enter the appropriate imaginary part of the coefficient as in 9.
   b. Press: CONT
   c. Go to 9 if there are more coefficients to be entered.
11. When "CHANGES (Y/N)?" is displayed:
   a. Enter Y if a change in the coefficients is desired.
   b. Press: CONT
   c. Go to 12.

   or

   a. Enter N if no changes in the coefficients are desired.
   b. Press: CONT
   c. Go to 15.
12. When "COEFFICIENT NUMBER?" is displayed:
   a. Enter the number of the coefficient to be changed.
   b. Press: CONT

13. When "Rcoef(I)?" (Where I is the number of the coefficient to be changed) is displayed:
    a. Enter the new value of Rcoef(I).
    b. Press: CONT
14. When "Icoef(I)?" (where I is the number of the coefficient to be changed) is displayed:
    a. Enter the new value of Icoef(I).
    b. Press: CONT
    c. Go to 11.
15. The real and imaginary parts of the roots will be printed. Any roots not found will contain the value 9.999999 E 99.

## EXAMPLE

```
DEGREE OF POLYNOMIAL= 4
MAX # OF ITERATIONS= 20
TOLERANCE FOR ROOTS= .00000001
TOLERANCE FOR FUNCTIONAL EVALUATIONS= .000001


COEFFICIENTS: [(Rcoef(0)+Icoef(0)*I)+(Rcoef(1)+Icoef(1)*I)*X^1+...]

         REAL              IMAGINARY

   -1.600000E+01        0.000000E+00
    0.000000E+00        0.000000E+00
    0.000000E+00        0.000000E+00
    0.000000E+00        0.000000E+00
    1.000000E+00        0.000000E+00




ROOTS:
         REAL              IMAGINARY


    2.000000E+00        0.000000E+00
    2.000000E+00        0.000000E+00
    0.000000E+00       -2.000000E+00
    0.000000E+00        2.000000E+00
```

## Driver Listing

```
10   ! ***********************************
20   ! *** DRIVER FOR SUBPROGRAM Siljak.
30   ! *** ROOTFINDER FOR POLYNOMIALS.
40   ! ***********************************
50   !
60   !
70   ! *** LINK ON Siljak AFTER DRIVER.
80   LINPUT "HAS Siljak ALREADY BEEN LINKED ON (Y/N)?",C$
90   IF (C$="Y") OR (C$="y") THEN 160
100  IF (C$="N") OR (C$="n") THEN 120
110  GOTO 80
120  LINK "Siljak",1010,160
130  !
140  !
150  ! *** INPUT DEGREE OF POLYNOMIAL.
160  INPUT "DEGREE OF POLYNOMIAL?",N
170  PRINT LIN(2),"DEGREE OF POLYNOMIAL=";N
180  IF N<=0 THEN 160
190  !
200  !
210  ! *** CALL Siljak1 TO DYNAMICALLY SET UP ARRAYS.
220  CALL Siljak1(N)
230  END
240  !
250  !
260  !
270  SUB Siljak1(N)
280  !
290  ! ******************************************************
300  ! *** Siljak1 IS USED TO DYNAMICALLY SET UP ARRAYS.
310  ! ******************************************************
320  !
330  !
340  DIM Rroot(1:N),Iroot(1:N),Rcoef(0:N),Icoef(0:N)
350  ! *** INPUT REMAINING PARAMETERS.
360  INPUT "MAX # OF ITERATIONS?",Itmax
370  PRINT "MAX # OF ITERATIONS=";Itmax
380  INPUT "TOLERANCE FOR ROOTS?",Tola
390  PRINT "TOLERANCE FOR ROOTS=";Tola
400  INPUT "TOLERANCE FOR FUNCTIONAL EVALUATIONS?",Tol
410  PRINT "TOLERANCE FOR FUNCTIONAL EVALUATIONS=";To
420  ! *** USE LOOP TO INPUT REAL AND IMAGINARY COEFF
430  PRINT LIN(2),"COEFFICIENTS: [(Rcoef(0)+Icoef(0)*.       .(1)+Icoe
f(1)*I)*X^1+...]"
440  PRINT LIN(1),SPA(8),"REAL","   IMAGINARY",LIN(1)
450  FOR I=0 TO N
460     DISP "Rcoef(";I;")=";
```

```
470       INPUT Rcoef(I)
480       DISP "Icoef(";I;")=";
490       INPUT Icoef(I)
500       PRINT USING 520:Rcoef(I),Icoef(I)
510    NEXT I
520    IMAGE 3X,MZ.6DE,5X,MZ.6DE
530    ! *** MAKE ANY NECESSARY CHANGES TO THE INPUT DATA.
540    LINPUT "CHANGES (Y/N)",C$
550    IF (C$="N") OR (C$="n") THEN 670
560    IF (C$="Y") OR (C$="y") THEN 580
570    GOTO 540
580    INPUT "COEFFICIENT NUMBER",I
590    IF (I<0) OR (I>N) THEN 580
600    DISP "Rcoef(";I;")=";
610    INPUT Rcoef(I)
620    DISP "Icoef(";I;")=";
630    INPUT Icoef(I)
640    PRINT USING 650;I,Rcoef(I),I,Icoef(I)
650    IMAGE "Rcoef(",DDD,")=",MZ.6DE,5X,"Icoef(",DDD,")=",MZ.6DE
660    GOTO 540
670    PRINT LIN(2)
680    !
690    !
700    ! *** ALL PARAMETERS HAVE BEEN ESTABLISHED.  CALL Siljak.
710    CALL Siljak(N,Rcoef(*),Icoef(*),Tolo,Tolf,Itmax,Rroot(*),Iroot(*)
)
720    !
730    !
740    ! *** PRINT REAL AND IMAGINARY ROOTS OF THE POLYNOMIAL.
750    PRINT LIN(2),"ROOTS:",LIN(1),SPA(8),"REAL","   IMAGINARY",LIN(2)
760    FOR I=1 TO N
770       PRINT USING 790;Rroot(I),Iroot(I)
780    NEXT I
790    IMAGE 3X,MZ.6DE,5X,MZ.6DE
800    PRINT LIN(5)
810    SUBEXIT
820    SUBEND
```

HEWLETT hp PACKARD

ROOTFINDER: BISECTION METHOD

Subprogram Name: Bisect

This program will search for solutions of F(x) = ∅ over an interval [a,b] where the user defines the continuous real valued function F(x). The function may be algebraic of the form $(a_0+a_1x^{e_1}- a_2x^{e_2}+...+ a_nx^{e_n})$ with $a_i$ real and $e_i$ rational (e.g., $x^3+ 3x^{3/2}+ \sqrt{x}$) or transcendental (e.g., sin(x) + cos(x)).

The user specifies the initial domain value, search increment, error tolerance, and maximum interval-halvings to be used.

Subprogram Utilization:

File Name: Bisect

Calling Syntax: CALL Bisect (A, B, Maxbi, Tol, Deltax, Root(*), F(*), Err(*), Nroots)

Input Parameters:

A - Lower bound of search interval.

B - Upper bound of search interval.

Maxbi - Maximum number of bisections for each subinterval .

Tol - Error tolerance for a root; a root x is accepted if $|F(x)| < Tol\ MAX(1,x)$

Deltax - Search increment; the subprogram begins at the lower bound and compares functional values at the ends of each subinterval, [a+ iΔx, a + (i+1)Δx], for a change of sign.

Nroots - Number of roots to be found.

Subprogram Required:

FNF(X) - Given a domain value x, this function subprogram should return the range value of the user-defined function; this function subprogram must be supplied

by the user.

Output Parameters:

Root(*) - Vector containing the roots of the function; the value 9.999999 E 99 is supplied for any roots not located.

F(*) - Vector containing the functional evaluations of the roots; again, the value 9.999999 E 99 is supplied for any roots not found.

Err(*) - Vector containing the estimated accuracy of the roots; the value 9.999999 E 99 is supplied for any roots not found.

Local Variables:

Baddta - Used in BAD DATA CHECK
Baddta = 1 implies bad data
Baddta = Ø implies reasonable data

C - # of iterations in a particular bracketed interval

Left - Left (lower) bound

N - Number of roots found

Noroot - Local equivalent of Nroots - # of roots to be found

Prod - Product of Y*F - check to determine when a root has been bracketed.

Right - Right (upper) bound

Size - Size of bisected interval

X - Estimate of root

Y,F - Functional evaluations of different X values.

Special Considerations and Programming Hints:

1. Because the number of roots located cannot be computed ahead of time, all roots are initialized to 9.999999 E 99 at the beginning of the subprogram. Thus on output, any roots not found will contain the value 9.999999 E 99.

2. One of the advantages of this method is that whenever a root is found, both the accuracy of the root (contained in array ERR(*)) and the accuracy of the functional evaluation of the root (contained in array F(*)) are known precisely.
3. Clearly, with enough effort, one can always locate a root to any desired accuracy with this subprogram. However, the bisection method converges rather slowly since it does not make full use of the information about F(x) available at each step. So, when possible, this method should not be used if there are a large number of roots to be found or if the rootfinder subprogram is to be called a number of times.
4. Upon entry into the subprogram, there is a bad data check. If the subprogram detects "nonsense" data, the following error message is printed and the program pauses:

```
"ERROR IN SUBPROGRAM Bisect."
A =          , B =          , Maxbi =

Tol=         , Deltax=      , Nroots=
```

The user may correct the data from the keyboard (e.g., Tol= 1 E-6, EXECUTE). When CONT is pressed, the program will resume execution at the next line.

5. If, in a particular subinterval, the maximum number of iterations is exceeded, the subprogram will print the following warning message and the approximate root and accuracies so far obtained:

```
"MAX # OF ITERATIONS REACHED ON ROOT # —
  "X BETWEEN" Left endpoint "AND" right endpoint.
  "F(X)=" Y.
"ACCURACY TO" —
"AVERAGE VALUE STORED IN Root(*) AS APPROXIMATE X."
```

The subprogram will then move to the next search interval.

6. System Configuration:
Standard Memory Option
(Optional) Printer

## Annotated Listing of Subprogram Bisect

```
10   SUB Bisect(A,B,Maxbi,Tol,Deltax,Root(*),F(*),Err(*),Nroots)
20   !
30   ! ************************************
40   ! *** ROOTFINDER - BISECTION METHOD.
50   ! ************************************
60   !
70   !
80   OPTION BASE 1
90   ! *** BAD DATA CHECK.
100  Baddta=(A>=B) OR (Maxbi<=0) OR (Tol<=0) OR (Deltax<=0) OR (Nroots
<=0)
110  IF Baddta=0 THEN 220
120  ! *** PRINT ERROR MESSAGE AND PAUSE.
130  ! *** USER MAY CORRECT DATA AND CONTINUE.
140  PRINT LIN(2),"ERROR IN SUBPROGRAM Bisect."
150  PRINT "A=";A;"  B=";B;"  Maxbi=";Maxbi
160  PRINT "Tol=";Tol;"  Deltax=";Deltax;"  Nroots=";Nroots,LIN(2)
170  PAUSE
171  GOTO 100
180  !
190  !
200  ! *** BEGIN SUBPROGRAM.
210  ! *** INITIALIZE LOCAL VARIABLES AND SET ALL ROOTS TO 9.999999E99
.
220  N=0
230  Noroot=Nroots
240  MAT Root=(9.999999E99)
250  MAT F=(9.999999E99)
260  MAT Err=(9.999999E99)
270  ! *** FUNCTIONAL VALUE OF LEFT BOUND.
280  X=A
290  ! *** IF DESIRED # OF ROOTS HAVE BEEN FOUND, SUBEXIT.
300  IF N>=Noroot THEN SUBEXIT
310  !
320  !
330  ! *** SEARCH FOR NEW ROOT.
340  N=N+1
350  Y=FNF(X)
360  F=Y
370  ! *** ADVANCE TO NEXT SEARCH INTERVAL.
380  A=A+Deltax
390  ! *** IF GREATER THAN UPPER BOUND THEN SUBEXIT.
400  IF A>B THEN SUBEXIT
410  X=A
420  Y=FNF(X)
430  Prod=F*Y
440  !
```

```
450  !
460  ! *** IF PRODUCT IS POSITIVE, SEARCH NEXT INTERVAL.
470  ! *** IF PRODUCT IS NEGATIVE, LOOK FOR ROOT.
480  IF Prod>0 THEN 360
490  IF Prod<0 THEN 650
500  IF F<>0 THEN 540
510  ! *** EXACT ROOT HAS BEEN FOUND. F(X)=0.
520  X=A-Deltax
530  Y=F
540  Root(N)=X
550  F(N)=Y
560  A=A+Deltax
570  Size=1E-12
580  Err(N)=Size
590  !
600  !
610  ! *** SEARCH NEXT INTERVAL FOR REMAINING ROOTS.
620  GOTO 280
630  ! *** ROOT HAS BEEN BRACKETED.
640  ! *** COMPUTE MIDPOINT AND ITS FUNCTIONAL VALUE.
650  Left=A-Deltax
660  Right=A
670  ! *** C IS NUMBER OF ITERATIONS.
680  C=0
690  X=(Left+Right)/2
700  Y=FNF(X)
710  C=C+1
720  !
730  !
740  ! *** CHECK FOR MAXIMUM # OF ITERATIONS.
750  ! *** IF MAXIMUM EXCEEDED THEN PRINT MESSAGE.
760  IF C>Maxbi THEN 1030
770  ! *** CHECK IF ERROR TOLERANCE IS SATISFIED.
780  IF ABS(Y)<Tol*MAX(1,X) THEN 940
790  !
800  !
810  ! *** IF PRODUCT IS POSITIVE, LOOK ON RIGHT INTERVAL.
820  Prod=F*Y
830  IF Prod<=0 THEN 870
840  Left=X
850  GOTO 690
860  ! *** CHECK IF ROOT HAS BEEN FOUND.
870  IF Prod=0 THEN 940
880  ! *** SEARCH ON LEFT INTERVAL.
890  Right=X
900  GOTO 690
910  !
```

```
920  !
930  ! *** ROOT HAS BEEN FOUND.
940  Root(N)=X
950  F(N)=Y
960  Size=Right-Left
970  Err(N)=Size
980  GOTO 280
990  !
1000 !
1010 ! *** MAX # OF ITERATIONS EXCEEDED. PRINT WARNING MESSAGE.
1020 ! *** PRINT APPROXIMATE ROOT AND ACCURACY.
1030 PRINT LIN(2),"MAX # OF BISECTIONS REACHED ON ROOT #";N
1040 PRINT "X BETWEEN ";Left;" AND ";Right
1050 PRINT "F(X)=";Y
1060 Size=Right-Left
1070 PRINT "ACCURACY TO ";Size
1080 PRINT "AVERAGE VALUE STORED IN Root(*) AS APPROXIMATE X",LIN(2)
1090 Root(N)=(Left+Right)/2
1100 F(N)=Y
1110 Err(N)=Size
1120 ! *** SEARCH NEXT INTERVAL FOR REMAINING ROOTS.
1130 GOTO 280
1140 SUBEND
```

Methods and Formulae:

This subprogram "Bisect" will search for solutions of $F(*)=\emptyset$ over an interval [a,b] where the user defines the real valued continuous function F(x). The function may be algebraic of the form $a_0 + a_1 x^{e1} + a_2 x^{e2} + \ldots + a_n x^{en}$ with $a_i$ real and $e_i$ rational (e.g. $x^3 + 3x^{3/2} + \sqrt{x}$) or transcendental (e.g., sin (x)+ cos (x)).

The user specifies the search increment $\Delta x$ and the error tolerance for F(x). The program then begins at the left of the interval and compares functional values at the ends of the subinterval [a, a+$\Delta$x]. If the functional values are of opposite sign then the bisection method is used to locate the root. Each subinterval [a + i $\Delta$x, a + (i+1) $\Delta$x] is examined for a possible root. At most one root per interval will be located and if there are multiple roots per interval, none may be located. The user must also specify a maximum number of interval-halvings (Maxbi), so that an error tolerance that is not satisfied will result in the root localized to an interval of size $2^{-Maxbi}$ (b-a). The subprogram will examine N = int $(\frac{b-a}{\Delta x})$ intervals.

REFERENCES:

1. Stark, Peter A., Introduction to Numerical Methods (London: MacMillan Company, Collier-MacMillan Limited, 1970), pp. 95-96.

Driver Utilization:

File Name: BISECT

The driver "BISECT" sets up the necessary input parameters for the subprogram "Bisect" as well as prints out the resulting outputs.

With each question, the user is only required to input the appropriate response. The driver also sets up the user-defined function subprogram.

User Instructions:

1. Insert the Utility Routines cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "BISECT: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "HAS Bisect ALREADY BEEN LINKED ON (Y/N)?" is displayed:
   a. Enter Y if the subprogram has already been linked on.
   b. Press: CONT
   c. Go to 5.

   or

   a. Enter N if the subprogram has not yet been linked on.
   b. Press: CONT
   c. At this point, Bisect,contained in File "Bisect", will be linked on after the driver.
   d. Go to 5.
5. When "HAS FUNCTION ALREADY BEEN DEFINED (Y/N)?" is displayed:
   a. Enter Y if the function has already been defined on the previous run.
   b. Press: CONT
   c. Go to 7.

   or

a. Enter N if the function has not yet been defined.
b. Press: CONT
c. Go to 6.

6. When "FUNCTION [e.g., F=(X-3)*(X+4)?" is displayed:
   a. Type in the user-defined function in the above format.
   b. Press: CONT
   c. At this point, the user-defined function will be properly entered in function subprogram FNF(X). This subprogram will then be stored on the mass storage device in file "KRYWEN" and then linked on to the program after subprogram Bisect. ("KRYWEN" must have been created previously to running this program.) Statement: CREATE "KRYWEN", 4, 84 does this. File "KRYWEN" has already been created on the Utility Library cartridge 1.
   d. Go to 7.
7. When "# OF ROOTS TO BE FOUND?" is displayed:
   a. Enter the number of roots to be found.
   b. Press: CONT
8. When "LOWER BOUND?" is displayed:
   a. Enter the lower bound of the search interval.
   b. Press: CONT
9. When "UPPER BOUND?" is displayed:
   a. Enter the upper bound of the search interval.
   b. Press: CONT
10. When "MAXIMUM # OF BISECTIONS?" is displayed:
   a. Enter the maximum number of bisections allowed in searching for any one root in a subinterval.
   b. Press: CONT
11. When "ERROR TOLERANCE?" is displayed:
   a. Enter the error tolerance desired. Here $|F(x)| \leq Tol*MAX(1,x)$ is the tolerance criterion.
   b. Press: CONT
12. When "SEARCH INCREMENT?" is displayed:
   a. Enter the search increment; i.e., the subprogram begins at the lower bound and compares functional values at the ends of each subinterval, $[a+i\Delta x, a+(i+1)\Delta x]$, for a change of sign.
   b. Press: CONT

13. The roots x, their functional values F(x), and the accuracy to which the root is found will be printed for all roots found over the interval. The value 9.999999 E 99 is inserted for any roots not found.

## EXAMPLE

```
F=SIN(X)-COS(X)/(1+X*X)


# OF ROOTS TO BE FOUND= 4
LOWER BOUNDS= 0
UPPER BOUNDS= 10
MAXIMUM # OF BISECTIONS= 20
ERROR TOLERANCE= .00000001
SEARCH INCREMENT= .1
```

| ROOT | FUNCTION VALUE | ACCURACY |
|---|---|---|
| 6.238996E-01 | -9.425000E-09 | 1.907350E-07 |
| 3.228892E+00 | 1.621320E-08 | 1.907400E-07 |
| 6.307698E+00 | -1.625780E-08 | 1.907000E-07 |
| 9.435884E+00 | 8.955260E-08 | 3.051750E-06 |

## Driver Listing

```
10   ! ******************************************
20   ! *** DRIVER FOR SUBPROGRAM Bisect.
30   ! *** ROOTFINDER USING THE BISECTION METHOD.
40   ! ******************************************
50   !
60   !
70   OPTION BASE 1
80   DIM A$(4)[84]
90   !
100  ! *** LINK Bisect ON AFTER DRIVER.
110  LINPUT "HAS Bisect ALREADY BEEN LINKED ON (Y/N)?",C$
120  IF (C$="Y") OR (C$="y") THEN 200
130  IF (C$="N") OR (C$="n") THEN 150
140  GOTO 110
150  LINK "Bisect",1010,200
160  !
170  !
180  ! *** SET UP FILE AND FUNCTION STRING FOR STORING AND RETRIEVING
190  ! *** USER-DEFINED FUNCTION.
200  LINPUT "HAS FUNCTION ALREADY BEEN DEFINED (Y/N)?",C$
210  IF (C$="Y") OR (C$="y") THEN 410
220  IF (C$="N") OR (C$="n") THEN 240
230  GOTO 200
240  ASSIGN #1 TO "KRYWEN"
250  A$(1)="10 DEF FN F(X)"
260  A$(2)="20 "
270  A$(3)="30 RETURN F"
280  A$(4)="40 FNEND"
290  ! *** INPUT USER-DEFINED FUNCTION.
300  LINPUT "FUNCTION [e.g., F=(X-3)*(X+4)]?",A$(2)[4]
310  PRINT LIN(2),A$(2)[4]
320  ! *** PRINT FUNCTION STRING ON FILE "KRYWEN".
330  FOR I=1 TO 4
340     PRINT #1,I;A$(I)
350  NEXT I
360  ! *** PLACE FUNCTION STRING AT END OF SUBPROGRAM.
370  LINK "KRYWEN",8000,410
380  !
390  !
400  ! *** INPUT NUMBER OF ROOTS TO BE LOCATED.
410  INPUT "# OF ROOTS TO BE FOUND?",Nroots
420  PRINT LIN(2),"# OF ROOTS TO BE FOUND=";Nroots
430  IF Nroots<=0 THEN 410
440  !
450  !
460  ! *** CALL Bisect1 TO DYNAMICALLY SET UP ARRAYS.
470  CALL Bisect1(Nroots)
```

```
480  END
490  !
500  !
510  !
520  SUB Bisect1(Nroots)
530  !
540  ! ************************************************
550  ! *** Bisect1 IS USED TO DYNAMICALLY SET UP ARRAYS.
560  ! ************************************************
570  !
580  !
590  OPTION BASE 1
600  DIM Root(Nroots),F(Nroots),Err(Nroots)
610  !
620  !
630  INPUT "LOWER BOUNDS?",A
640  PRINT "LOWER BOUNDS=";A
650  INPUT "UPPER BOUNDS?",B
660  PRINT "UPPER BOUNDS=";B
670  INPUT "MAXIMUM # OF BISECTIONS?",Maxbi
680  PRINT "MAXIMUM # OF BISECTIONS=";Maxbi
690  INPUT "ERROR TOLERANCE?",Tol
700  PRINT "ERROR TOLERANCE=";Tol
710  INPUT "SEARCH INCREMENT?",Deltax
720  PRINT "SEARCH INCREMENT=";Deltax
730  !
740  !
750  ! *** ALL PARAMETERS HAVE BEEN ESTABLISHED. CALL Bisect.
760  CALL Bisect(A,B,Maxbi,Tol,Deltax,Root(*),F(*),Err(*),Nroots)
770  !
780  !
790  ! *** PRINT ROOTS, FUNCTION VALUES AND ACCURACY.
800  ! *** 9.999999E99 IS STORED IN ROOT(*) FOR ANY ROOTS NOT LOCATED.
810  PRINT LIN(2),SPA(5),"ROOT",SPA(3),"FUNCTION VALUE",SPA(9),"ACCURA
CY",LIN(1)
820  FOR I=1 TO Nroots
830     PRINT USING 840;Root(I),F(I),Err(I)
840     IMAGE    MZ.6DE,10X,MZ.6DE,10X,MZ.6DE
850  NEXT I
860  SUBEXIT
870  SUBEND
```

Subprogram Name: Spline

This subprogram computes a curve s(x) that passes through the N data points $(x_i, y_i)$ supplied by the user and computes certain information at any point $t_j$ on the curve as long as $t_j$ is in the interval $[x_1, x_n]$. The information that can be computed is the integral over the interval and the derivative or functional value at any point on the interval.

Subprogram Utilization:

File Name: Spline
Calling Syntax:

Input Parameters:

N - Number of data points.

Narg - Number of arguments for which the derivative or functional value is to be computed; Narg = ∅ means that only the integral is to be calculated.

X(*) - Vector containing domain values of the data points subscripted from 1 to N; the X(i) values should be entered in increasing order.

Y(*) - Vector containing range values of the data points subscripted from 1 to n.

Domain(*) - Vector containing the domain values for which the derivative or functional value is to be computed; subscripted from 1 to Narg; arguments may be entered in any order, but these values must be contained in the interval [X(1),X(N)].

Eps - Error tolerance in iterative solution of simultaneous equations.

Output Parameters:

Func(*) - Vector containing the interpolated function values for output arguments in Domain(*); subscripted from 1 to Narg.

Deriv(*) - Vector containing the derivative values for output arguments in Domain(*); subscripted from 1 to Narg.

Int - Integral $\int_{X_1}^{Xn} s(x)dx$.

Local Variables:

Baddta - Used in BAD DATA CHECK
Baddta = 1 implies bad data
Baddta = Ø implies reasonable data

H - Temporary storage for $x_{i+1} - x_{i-1}$, $|T|$, $x_{i+1} - x_i$ or $t_j - x_i$.

G(*) - 3T

I,J - Loop counter

S - Temporary storage for S(I)+H*G(I)

S(*) - 2T or $S''(t_j)$ vector of second derivative

T - $\dfrac{\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}}; \quad t_j - x_{i+1}}{H}$

U - Current maximum value of $|T|$ used to test error tolerance;

$$\frac{1}{6}(S''(x_i)+S''(x_{i+1}))+$$

$$S''(x_i) + 3\,T\,(t_j - x_i)$$

W - $8-4\sqrt{3}$; successive over-relaxation factor used in iterative solution; $\frac{y_{i+1} - y_i}{x_{i+1} - x_i}$.

X - Temporary storage for $x_i - x_{i-1}$.

Xi - $x_i$

Ximl - $x_{i-1}$

Xipl - $x_{i+1}$

Yi - $y_i$

Yiml - $y_{i-1}$

Yipl - $y_{i+1}$

Special Considerations and Programming Hints:

1. Upon entry into the subprogram, there is a bad data check. If the subprogram detects "nonsense" data, the following error message is printed and the program pauses:

   "ERROR IN SUBPROGRAM SPLINE."
   N = , Eps =

   The data may be corrected from the keyboard (e.g., Eps = 1 E-6, EXECUTE). When CONT is pressed, the program will resume execution at the next line.

2. The arguments for which the derivative or functional value are to be computed must lie in the interval [X(1),X(N)]. If an argument is found outside this range, the following error message is printed and the program pauses:

   "ERROR IN SUBPROGRAM SPLINE."
   "ARGUMENT OUT OF BOUNDS."
   X(1) = , X(N) = , Domain (I) =

   The program may be saved in the following way:
   a. Type: Domain (I) = a permissible value
   b. Press: EXECUTE
   c. Type: CONT Corrector. (Type each letter. Do not use the CONT key.)
   d. Press: EXECUTE

3. The data points $(X_i, Y_i)$ i=1, n should be entered in increasing order of the $X_i$, where the $X_i$ are discrete and $X_i < X_{i+1}$ for i = 1, n-1. The output arguments $t_j$, j = 1, Narg may be entered in any order.

4. The error factor of the solutions is approximately equal to $h^4$ for the integral, $h^3$ for the functional values and $h^2$ for the derivative, where h is the average interval size.
5. X(*), Y(*), Domain(*), Deriv(*) and Func(*) must be dimensioned in the calling program.
6. System Configuration:
   Standard Memory Option
   (Optional) Printer

# Annotated Listing of Subprogram Spline

```
10    SUB Spline(N,Narg,X(*),Y(*),Domain(*),Func(*),Deriv(*),Int,Eps)
20    !
30    ! ***************************************************************
*****
40    ! *** SPLINE FIT FOR FUNCTION VALUES, INTEGRATION AND DIFFERENTIA
TION.
50    ! ***************************************************************
*****
60    !
70    !
80    ! *** BAD DATA CHECK.
90    Baddta=(N<=0) OR (Eps<=0)
100   IF Baddta=0 THEN 200
110   ! *** PRINT ERROR MESSAGE AND PAUSE.
120   ! *** USER MAY CORRECT DATA AND CONTINUE.
130   PRINT LIN(2),"ERROR IN SUBPROGRAM Spline."
140   PRINT "N=";N,"Eps=";Eps,LIN(2)
150   PAUSE
160   GOTO 90
170   !
180   !
190   ! *** BEGIN SUBPROGRAM.
200   OPTION BASE 1
210   DIM S(N),G(N-1),Work(N-1)
220   FOR I=2 TO N-1
230      Xi=X(I)
240      Xim1=X(I-1)
250      Xip1=X(I+1)
260      Yi=Y(I)
270      Yim1=Y(I-1)
280      Yip1=Y(I+1)
290      X=Xi-Xim1
300      H=Xip1-Xim1
310      Work(I)=.5*X/H
320      T=((Yip1-Yi)/(Xip1-Xi)-(Yi-Yim1)/X)/H
330      S(I)=2*T
340      G(I)=3*T
350   NEXT I
360   S(1)=S(N)=0
370   ! *** W IS THE RELAXATION FACTOR FOR SUCCESSIVE OVER-RELAXATION.
380   W=8-4*SQR(3)
390   U=0
400   FOR I=2 TO N-1
410      T=W*(-S(I)-Work(I)*S(I-1)-(.5-Work(I))*S(I+1)+G(I))
420      H=ABS(T)
430      IF H>U THEN U=H
440      S(I)=S(I)+T
```

```
450   NEXT I
460   IF U>=Eps THEN 390
470   FOR I=1 TO N-1
480      G(I)=(S(I+1)-S(I))/(X(I+1)-X(I))
490   NEXT I
500   IF Narg=0 THEN 820
510   !
520   !
530   ! *** CALCULATE FUNCTION VALUES AND DERIVATIVES.
540   FOR J=1 TO Narg
550 Corrector:      I=1
560      T=Domain(J)
570      IF T>=X(1) THEN 650
580      PRINT LIN(2),"ERROR IN SUBPROGRAM Spline."
590      PRINT "ARGUMENT OUT OF BOUNDS."
600      PRINT "X(1)=";X(1),"X(N)=";X(N),"Domain(";J;")=";Domain(J),LIN
(2)
610      PAUSE
620      GOTO 550
630      !
640      !
650      I=I+1
660      IF I>N THEN 580
670      IF T>X(I) THEN 650
680      I=I-1
690      H=Domain(J)-X(I)
700      T=Domain(J)-X(I+1)
710      X=H*T
720      S=S(I)+H*G(I)
730      Z=1/6
740      U=Z*(S(I)+S(I+1)+S)
750      W=(Y(I+1)-Y(I))/(X(I+1)-X(I))
760      Func(J)=W*H+Y(I)+X*U
770      Deriv(J)=W+(H+T)*U+Z*X*G(I)
780   NEXT J
790   !
800   !
810   ! *** CALCULATE INTEGRAL FROM X(1) TO X(N).
820   Int=0
830   FOR I=1 TO N-1
840      H=X(I+1)-X(I)
850      Int=Int+.5*H*(Y(I)+Y(I+1))-1/24*H^3*(S(I)+S(I+1))
860   NEXT I
870   SUBEND
```

Method and Formulae:

This subprogram computes a curve s(x) that passes through the n data points $(x_i, y_i)$ supplied by the user and computes certain information at any point $t_j$ on the curve as long as $t_j$ is in the interval $[x_1, x_n]$. The information that can be computed is the integral over the interval and the derivative or functional value at any point on the interval.

The method implemented fits a curve through the points and integrates, differentiates and interpolates that curve. The curve used is the cubic natural spline which derives its name from a draftsman's mechanical spline. If the spline is considered as a function represented by s(x), then the second derivative s''(x) approximates the curvature. For the curve through data points $(x_1, y_1)$, $(x_2, y_2)$, ... , $(x_n, y_n)$ we want $\int_{x_1}^{x_n} (s''(x))^2 \, dx$ to be minimized in order to achieve the "smoothest" curve.

The spline function with minimum curvature has cubic polynomials between adjacent data points. Adjacent polynomials are joined continuously with continuous first and second derivatives as well as $s''(x_1) = s''(x_n) = 0$.

The procedure to determine s(x) involves the iterative solution of a set of simultaneous linear equations by Young's method of successive over-relaxation. The accuracy to which these equations are solved is specified by the user.

The formulae employed are:

$$\int_{x_i}^{x_{i+1}} s(x)dx \approx \{\frac{1}{2} (x_{i+1} - x_i) (y_i + y_{i+1}) - \frac{1}{24} (x_{i+1} - x_i)^3 [s''(x_i) + s''(x_{i+1})]\}$$

$$s(t_j) = y_i + (t_j - x_i)(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}) + (t_j - x_i) (t_j - x_{i+1}) \frac{1}{6} [s''(x_i) + s''(x_{i+1}) + s''(t_j)]$$

$$s'(t_j) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{1}{6}[(t_j - x_i) + (t_j - x_{i+1})](s''(x_i) +$$

$$s''(t_j)) + \frac{1}{6}(t_j - x_i)(t_j - x_{i+1})\left(\frac{s''(x_{i+1}) - s''(x_i)}{x_{i+1} - x_i}\right)$$

REFERENCES:


1. Ralston and Wilf, Mathematical Methods for Digital Computers, Vol. II (New York: John Wiley and Sons, 1967) pp. 156-158.

2. Greville, T.N.E., Editor, "Proceedings of An Advanced Seminar Conducted by the Mathematics Research Center", U.S. Army, University of Wisconsin, Madison. October 7-9, 1968. Theory and Applications of Spline Functions (New York, London: Academic Press, 1969), pp. 156-167.

Driver Utilization:

File Name: SPLINE

The driver "SPLINE" sets up the necessary input parameters for the subprogram Spline as well as prints the resulting outputs.

With each question, the user is only required to respond with the appropriate answer.

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SPLINE: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "HAS Spline ALREADY BEEN LINKED ON (Y/N)?" is displayed:
   a. Enter:Y if the subprogram has already been linked on.
   b. Press: CONT
   c. Go to step 5.

   or

   a. Enter: N if the subprogram has not yet been linked on
   b. Press: CONT
   c. At this point, subprogram Spline contained in file "Spline" is linked on after the driver.
   d. Go to step 5.
5. When "# OF DATA POINTS?" is displayed:
   a. Enter the number of data points to be used in the calculation.
   b. Press: CONT
6. When "# OF DOMAIN POINTS?" is displayed:
   a. Enter the number of arguments for which the derivative or interpolated functional value is to be computed. If only the value of the integral is desired, enter Ø.
   b. Press: CONT

7. At this point, subprogram Splinel is called to dynamically set up the arrays X(*), Y(*), Domain(*), Deriv(*) and Func(*).
8. When "ERROR TOLERANCE?" is displayed:
   a. Enter the error tolerance for the iterative solution of the simultaneous equations used in the subprogram.
   b. Press: CONT
9. When "DATA POINTS:" is printed and "X(I)?" (For I=1 to # of Data Points) is displayed:
   a. Enter the appropriate X-value. The data points should be entered so that the X(I) are discrete and X(I)<X(I+1) for I = 1 to N-1. The points need not be equally spaced.
   b. Press: CONT
10. When "Y(I)?" (For I=1 to # of Data Points) is displayed:
   a. Enter the appropriate Y-value.
   b. Press: CONT
   c. Go to 9 if there are more data values to be entered. Otherwise, go to 11.
11. When "CHANGES (Y/N)?" is displayed:
   a. Enter Y if changes are to be made in the data.
   b. Press: CONT
   c. Go to 12.

   or

   a. Enter N if the data is correct.
   b. Press: CONT
   c. Go to 15.
12. When "DATA POINT #?" is displayed:
   a. Enter the number of the data point to be changed.
   b. Press: CONT
13. When "X(I)?" (where I is the number of the data point to be changed) is displayed:
   a. Enter the correct X-value.
   b. Press: CONT
14. When "Y(I)?" (where I is the number of the data point to be changed) is displayed:
   a. Enter the correct Y-value.
   b. Press: CONT

c. Go to 11.

15. When "DOMAIN VALUES:" is printed and "Domain(I)?" (For I=1 to number of domain points) is displayed:
    a. Enter the appropriate X-value to be evaluated in the cubic spline. Domain values must be contained in the interval [X(1), X(N)].
    b. Press: CONT
    c. Go to 15 if there are more domain values to be entered, otherwise go to 16.

16. When "CHANGES (Y/N)?" is displayed:
    a. Enter Y if changes are to be made in the data.
    b. Press: CONT
    c. Go to 17

    or

    a. Enter N if the data is correct.
    b. Press: CONT
    c. Go to 19.

17. When "DOMAIN VALUE #?" is displayed:
    a. Enter the number of the domain value to be changed.
    b. Press: CONT

18. When "DOMAIN(I)?" (where I is the number of the domain value to be changed) is displayed:
    a. Enter the correct domain value.
    b. Press: CONT
    c. Go to 16.

19. The program will print the resulting interpolated functional values, the values of the derivative and the value of the integral over the interval [X(1), X(N)].

```
# OF DATA POINTS= 5
# OF DOMAIN POINTS= 4
ERROR TOLERANCE= .000001


DATA POINTS:

   X( 1)=-3.000000E+00      Y( 1)= 3.000000E+00
   X( 2)=-1.000000E+00      Y( 2)= 6.000000E+00
   X( 3)= 2.000000E+00      Y( 3)= 8.000000E+00
   X( 4)= 4.000000E+00      Y( 4)= 2.000000E+00
   X( 5)= 7.000000E+00      Y( 5)= 5.000000E+00


DOMAIN VALUES:

   Domain( 1)=-2.500000E+00
   Domain( 2)= 0.000000E+00
   Domain( 3)= 2.500000E+00
   Domain( 4)= 5.000000E+00


INTEGRAL FROM -3.000000E+00 TO  7.000000E+00 =  4.984962E+01


DOMAIN VALUES     FUNCTION VALUES    DERIVATIVE VALUES

-2.500000E+00       3.691810E+00       1.399138E+00
 0.000000E+00       7.752235E+00       1.573946E+00
 2.500000E+00       6.670259E+00      -3.141092E+00
 5.000000E+00       1.343550E+00       5.030651E-01
```

## Driver Listing

```
10   ! ********************************************************
20   ! *** DRIVER FOR Spline - SPLINE FIT FOR INTERPOLATION,
30   ! *** INTEGRATION AND DIFFERENTIATION OF DATA POINTS.
40   ! ********************************************************
50   !
60   !
70   ! *** LINK ON Spline.
80   LINPUT "HAS Spline ALREADY BEEN LINKED ON (Y/N)?",C$
90   IF (C$="Y") OR (C$="y") THEN 160
100  IF (C$="N") OR (C$="n") THEN 120
110  GOTO 80
120  LINK "Spline",2010,160
130  !
140  !
150  ! *** DETERMINE ARRAY SIZES.
160  INPUT "# OF DATA POINTS?",N
170  PRINT LIN(2),"# OF DATA POINTS=";N
180  IF N<2 THEN 160
190  INPUT "# OF DOMAIN POINTS [i.e., # OF INTERPOLATION POINTS]?",Nar
g
200  PRINT "# OF DOMAIN POINTS=";Narg
210  IF Narg<0 THEN 190
220  ! *** SET Flg=1 IF Narg=0 AND THEN SET Narg=1, SO THAT ARRAYS MAY
 BE
230  ! *** DYNAMICALLY SET UP.
240  IF Narg=0 THEN Flg=1
250  IF Narg=0 THEN Narg=1
260  ! *** CALL Spline1 TO DYNAMICALLY SET UP ARRRAYS.
270  CALL Spline1(N,Narg,Flg)
280  END
290  !
300  !
310  !
320  SUB Spline1(N,Narg,Flg)
330  !
340  ! ************************************************************
******
350  ! *** THIS SUBPROGRAM IS USED SOLEY TO ESABLISH THE ARRAYS DYNAMI
CALLY.
360  ! ************************************************************
******
370  !
380  !
390  ! *** SET UP ARRAYS.
400  OPTION BASE 1
410  DIM X(N),Y(N),Domain(Narg),Deriv(Narg),Func(Narg)
420  IF Flg=1 THEN Narg=0
```

```
430  INPUT "ERROR TOLERANCE?",Eps
440  PRINT "ERROR TOLERANCE=";Eps
450  ! *** INPUT DATA POINTS.
460  PRINT LIN(2),"DATA POINTS:",LIN(1)
470  FOR I=1 TO N
480     FIXED 0
490     DISP "X(";I;")";
500     INPUT X(I)
510     DISP "Y(";I;")";
520     INPUT Y(I)
530     PRINT USING 540;I,X(I),I,Y(I)
540     IMAGE 3X,"X(",DD,")=",MZ.6DE,5X,"Y(",DD,")=",MZ.6DE
550  NEXT I
560  LINPUT "CHANGES (Y/N)?",C$
570  IF (C$="Y") OR (C$="y") THEN 600
580  IF (C$="N") OR (C$="n") THEN 720
590  GOTO 560
600  INPUT "DATA POINT #?",I
610  IF (I<=0) OR (I>N) THEN 600
620  FIXED 0
630  DISP "X(";I;")";
640  INPUT X(I)
650  DISP "Y(";I;")";
660  INPUT Y(I)
670  PRINT USING 540;I,X(I),I,Y(I)
680  GOTO 560
690  !
700  !
710  ! *** INPUT DOMAIN VALUES.
720  PRINT LIN(2),"DOMAIN VALUES:",LIN(1)
730  FOR I=1 TO Narg
740     DISP "Domain(";I;")";
750     INPUT Domain(I)
760     PRINT USING 770;I,Domain(I)
770     IMAGE 3X,"Domain(",DD,")=",MZ.6DE
780  NEXT I
790  LINPUT "CHANGES (Y/N)?",C$
800  IF (C$="Y") OR (C$="y") THEN 830
810  IF (C$="N") OR (C$="n") THEN 920
820  GOTO 790
830  INPUT "DOMAIN VALUE #?",I
840  IF (I<=0) OR (I>Narg) THEN 830
850  DISP "Domain(";I;")";
860  INPUT Domain(I)
870  PRINT USING 770;I,Domain(I)
880  GOTO 790
890  !
```

```
900  !
910  ! *** ALL PARAMETERS HAVE BEEN ESTABLISHED.  CALL Spline.
920  CALL Spline(N,Narg,X(*),Y(*),Domain(*),Func(*),Deriv(*),Int,Eps)
930  !
940  !
950  ! *** PRINT RESULTS.
960  PRINT USING 970;X(1),X(N),Int
970  IMAGE //,"INTEGRAL FROM ",MZ.6DE," TO ",MZ.6DE," = ",MZ.6DE
980  IF Flg=1 THEN 1040
990  PRINT LIN(2),"DOMAIN VALUES    FUNCTION VALUES   DERIVATIVE VALUE
S",LIN(1)
1000 FOR I=1 TO Narg
1010    PRINT USING 1020;Domain(I),Func(I),Deriv(I)
1020    IMAGE 3(MZ.6DE,5X)
1030 NEXT I
1040 SUBEND
```

HEWLETT hp PACKARD

# NUMERICAL INTEGRATION: SIMPSON'S RULE

Subprogram Name: Simp

This subprogram approximates $\int_a^b f(x)dx$ for the user-defined continous function f(x) on the interval [a,b]. The user is required to specify the maximum number of iterations permitted and the error tolerance between successive calculations of the integral.

Subprogram Utilization:

File Name: Simp

Calling Syntax: CALL Simp (Low, Up, Int, Flg, Itmax, Tol)

Input Parameters:

Low - Lower bound of interval of integration.

Up - Upper bound of interval of integration.

Flg - Should intermediate results be printed:

Flg = 1 implies yes

Flg = 1 implies no

Itmax - Maximum number of interval halvings

Tol - Error tolerance between successive calculations of integral.

Subprograms Required:

FNF(x)- Function to be integrated must be defined in function subprogram FNF(x).

Output Parameters:

Int - Value of integral.

Local Variables:

Baddta - Used in BAD DATA CHECK.

Baddta = 1 implies bad data.

Baddta = Ø implies data is reasonable.

Darg - Domain argument; temporary lower bound.
Frst - Used to eliminate the Tol stopping criterion on first computation of integral.
Intold - Previous value of integral used for stopping criterion.
N - Current number of intervals (N=2↑I).
Siz - Interval size (Up-low)/N.
Temp - Temporary storage location.
X - Domain argument for function evaluation.
Y - Functional value F(x).

Special Considerations and Programming Hints:

1. Upon entry into the subprogram, a BAD DATA CHECK is performed. If the program detects "nonsense" data, the program execution will pause and the following error message will be printed:

"ERROR IN SUBPROGRAM Simp."
Up = , Low = , Flg =

Itmax = , Tol =

The data may be corrected from the keyboard (e.g., Tol = 1E-4, EXECUTE). When CONT is pressed, the program will resume execution at the next line.

2. If the maximum number of iterations is exceeded, the following error message will be printed:

"MAXIMUM # OF ITERATIONS EXCEEDED."
Int = , Intold =
Tol = , Itmax =

At this point, the tolerance or the maximum number of iterations may be increased from the keyboard (e.g., Itmax = 3Ø, EXECUTE) or the value of the integral may be accepted as is.

3. System Configuration:
Standard Memory option
(Optional) Printer

# Annotated Listing of Subprogram Simp

```
10   SUB Simp(Low,Up,Int,Flg,Itmax,Tol)
20   !
30   ! ************************************************************
40   ! *** PERFORMS NUMERICAL INTEGRATION USING SIMPSON'S RULE.
50   ! ************************************************************
60   !
70   !
80   ! *** BAD DATA CHECK.
90   Baddta=(Up<=Low) OR (Flg=0) OR (Itmax<=0) OR (Tol<=0)
100  IF Baddta=0 THEN 210
110  ! *** PRINT ERROR MESSAGE AND PAUSE.
120  ! *** USER MAY CORRECT DATA AND CONTINUE.
130  PRINT LIN(2),"ERROR IN SUBPROGRAM Simp."
140  PRINT "Up=";Up;"  Low=";Low;"  Flg=";Flg
150  PRINT "Itmax=";Itmax;"  Tol=";Tol,LIN(2)
160  PAUSE
170  GOTO 90
180  !
190  !
200  ! *** BEGIN SUBPROGRAM.  INITIALIZE LOCAL VARIABLES.
210  Int=I=0
220  Frst=1
230  X=Low
240  Y=FNF(X)
250  Int=Int+Y
260  X=Up
270  Y=FNF(X)
280  Temp=Int=Int+Y
290  I=I+1
300  ! *** CHECK IF MAXIMUM NUMBER OF ITERATIONS EXCEEDED.
310  IF I<=Itmax THEN 430
320  ! *** MAXIMUM NUMBER OF ITERATIONS EXCEEDED.
330  ! *** PRINT ERROR MESSAGE AND PAUSE.
340  PRINT LIN(2),"ERROR IN SUBPROGRAM Simp."
350  PRINT "MAXIMUM # OF ITERATIONS EXCEEDED."
360  PRINT USING 370;Int,Intold,Tol,Itmax
370  IMAGE "Int=",MZ.6DE,5X,"Intold=",MZ.6DE,/,"Tol=",MZ.6DE,5X,"Itmax
=",DDD,//
380  PAUSE
390  !
400  !
410  ! *** COMPUTE NUMBER OF INTERVALS, N, INTERVAL SIZE, Siz, AND FIR
ST
420  ! *** INTERVAL VALUE, Y.  LOWER BOUND IS ASSIGNED TO Darg.
430  N=2^I
440  Siz=(Up-Low)/N
450  X=Siz+Low
```

```
460   Y=FNF(X)
470   Int=Int+4*Y
480   Darg=Low
490   ! *** INCREMENT Darg, THE TEMPORARY LOWER BOUND.
500   Darg=Darg+2*Siz
510   ! *** Darg IS INCREMENTED BY 2 INTERVAL SIZES UP TO THE
520   ! *** UPPER BOUND Up.
530   IF Darg<Up THEN 780
540   ! *** CALCULATE APPROXIMATION TO INTEGRAL.
550   Int=Siz*Int/3
560   ! *** CHECK IF INTERMEDIATE VALUES SHOULD BE PRINTED.
570   IF Flg=-1 THEN 610
580   ! *** PRINT INTERMEDIATE VALUES.
590   PRINT USING 600;N,Int
600   IMAGE "# INTERVAL=",DDDD,5X,"INTEGRAL=",MZ.6DE
610   IF Frst=0 THEN 650
620   Frst=0
630   GOTO 700
640   ! *** CHECK IF STOPPING CRITERION IS SATISFIED.
650   IF ABS(Intold-Int)>=Tol THEN 700
660   SUBEXIT
670   !
680   !
690   ! *** SAVE FORMER SUM, Int, IN Intold, FOR LATER CONVERGENCE CHEC
K.
700   Intold=Int
710   Int=Temp
720   ! *** HALF INTERVAL SIZE AGAIN.
730   GOTO 290
740   !
750   !
760   ! *** MULTIPLY SUCCESSIVE FUNCTION VALUES BY 2 OR 4 AND
770   ! *** UPDATE SUM.
780   X=Darg
790   Y=FNF(X)
800   Int=Int+2*Y
810   X=Darg+Siz
820   Y=FNF(X)
830   Int=Int+4*Y
840   ! *** INCREMENT Darg, THE TEMPORARY LOWER BOUND.
850   GOTO 500
860   SUBEND
```

Methods and Formulae:

This subprogram will approximate $\int_a^b F(x)dx$ for the user-defined function F(x) which must be defined in a function subprogram. The function must be continous over the interval [a,b].

The method used is Simpson's one-third rule with truncation error $O(h^4)$ where h is the interval size.

The stopping criterion for this method is either a maximum number of interval halvings or successive computations of the integral differing by less than some user-supplied error tolerance.

Simpson's one-third rule:

$$\int_a^b F(x)dx \frac{h}{3} [f(a)+4F(a+h)+2F(a+2h)+4F(a+3h)+...+ 4F(a+(n-1)h)+F(a+nh)]$$

where n = number of intervals,

$h = \frac{(b-a)}{n}$ = interval size.

REFERENCES:

1. Beckett, Royce and Hurt, James, NUMERICAL CALCULATIONS AND ALGORITHMS (New York: McGraw Hill, 1967), pp. 166-169.

Driver Utilization:

File Name: SIMP

The driver "SIMP" sets up the necessary input parameters for the subprogram Simp and prints the resulting value of the integral. Intermediate integral values may also be printed at the user's option.

With each question, the user is only required to enter the appropriate response. The driver also sets up the user-defined function subprogram.

User Instructions:

1. Insert the Utility Routines cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SIMP: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "HAS Simp ALREADY BEEN LINKED ON (Y/N)?" is displayed:
   a. Enter Y if the subprogram has already been linked on.
   b. Press: CONT
   c. Go to 5.

   or

   a. Enter N if the subprogram has not yet been linked on.
   b. Press: CONT
   c. At this point, subprogram Simp contained in File "Simp" will be linked on at the end of the driver.
   d. Go to 5.
5. When "HAS FUNCTION ALREADY BEEN DEFINED (Y/N)?" is displayed:
   a. Enter Y if the function has already been defined on the previous sum.
   b. Press: CONT
   c. Go to 7.

   or

   a. Enter N if the function has not yet been defined.
   b. Press: CONT
   c. Go to 6.

6. When "FUNCTION [e.g., F=(X-3)*(X+4)]?" is displayed:
   a. Type in the user-defined function to be integrated in the format F = ——.
   b. Press: CONT
   c. At this point, the user-defined function will be properly entered in function subprogram FNF(X). This subprogram will then be stored on the mass storage device in file "KRYWEN" and then linked on to the program after subprogram Simp. (File "KRYWEN" must have been created previously to running this program. Statement: CREATE "KRYWEN", 4,84 does this. File "KRYWEN" has already been created on the Utility Library cartridge 1.)
7. When "LOWER BOUND?" is displayed:
   a. Enter the lower bound of the interval.
   b. Press: CONT
8. When "UPPER BOUND?" is displayed:
   a. Enter the upper bound of the interval.
   b. Press: CONT
9. When "PRINT INTERMEDIATE DATA POINTS (Y/N)?" is displayed:
   a. Enter Y if intermediate data points are desired.
   b. Press: CONT
   c. Go to 1Ø.
   or
   a. Enter N if intermediate data points are not desired.
   b. Press: CONT
   c. Go to 1Ø.
10. When "MAX # OF INTERVAL HALVINGS?" is displayed:
   a. Enter the maximum number of interval halvings. The evaluation of the integral will be made on 2 subintervals, the 4, 16, 32, ..., halving the interval size on each iteration.
   b. Press: CONT
11. When "ERROR TOLERANCE?" is displayed:
   a. Enter the error tolerance. The value of the integral is accepted if the difference in value of two successive approximations is less than this tolerance.

b. Press: CONT

12. The program will print the value of the integral as well as intermediate data points if they were requested.

1. $\int_0^{\pi/3} x^2 \sin(3x)dx = 0.2173926$

```
F=X*X*SIN(3*X)


LOWER BOUND= 0
UPPER BOUND= 1.0471975512
MAX # OF INTERVAL HALVINGS= 10
ERROR TOLERANCE= .000001

# INTERVAL=    2      INTEGRAL= 1.913968E-01
# INTERVAL=    4      INTEGRAL= 2.170216E-01
# INTERVAL=    8      INTEGRAL= 2.173795E-01
# INTERVAL=   16      INTEGRAL= 2.173921E-01
# INTERVAL=   32      INTEGRAL= 2.173927E-01


INTEGRAL= 2.173927E-01
```

2. $\int_1^{10} \log(x)dx = 14.0258509$

```
F=LOG(X)


LOWER BOUND= 1
UPPER BOUND= 10
MAX # OF INTERVAL HALVINGS= 10
ERROR TOLERANCE= .00000001

# INTERVAL=    2      INTEGRAL= 1.368237E+01
# INTERVAL=    4      INTEGRAL= 1.396310E+01
# INTERVAL=    8      INTEGRAL= 1.401744E+01
# INTERVAL=   16      INTEGRAL= 1.402593E+01
# INTERVAL=   32      INTEGRAL= 1.402579E+01
# INTERVAL=   64      INTEGRAL= 1.402585E+01
# INTERVAL=  128      INTEGRAL= 1.402585E+01
# INTERVAL=  256      INTEGRAL= 1.402585E+01
# INTERVAL=  512      INTEGRAL= 1.402585E+01
# INTERVAL=1024      INTEGRAL= 1.402585E+01


INTEGRAL= 1.402585E+01
```

# Driver Listing

```
10   ! ********************************************************
20   ! *** DRIVER FOR SUBPROGRAM Simp -- SIMPSON'S METHOD FOR
30   ! *** NUMERICAL INTEGRATION.
40   ! ********************************************************
50   !
60   !
70   ! *** LINK ON SUBPROGRAM Simp AFTER USER-DEFINED SUBPROGRAM.
80   LINPUT "HAS Simp ALREADY BEEN LINKED ON (Y/N)?",C$
90   IF (C$="Y") OR (C$="y") THEN 170
100  IF (C$="N") OR (C$="n") THEN 120
110  GOTO 80
120  LINK "Simp",1010,170
130  !
140  !
150  ! *** SET UP FILE AND FUNCTION STRING FOR STORING AND RETRIEVING
160  ! *** USER-DEFINED FUNCTION.
170  OPTION BASE 1
180  DIM A$(4)[80]
190  LINPUT "HAS FUNCTION ALREADY BEEN DEFINED (Y/N)?",C$
200  IF (C$="Y") OR (C$="y") THEN 400
210  IF (C$="N") OR (C$="n") THEN 230
220  GOTO 190
230  ASSIGN #1 TO "KRYWEN"
240  A$(1)="10 DEF FN F(X)"
250  A$(2)="20   "
260  A$(3)="30 RETURN F"
270  A$(4)="40 FN END"
280  ! *** INPUT AND PRINT USER-DEFINED FUNCTION.
290  LINPUT "FUNCTION [e.g., F=(X-3)*(X+4)]?",A$(2)[4]
300  PRINT LIN(2),A$(2)[4]
310  ! *** PRINT FUNCTION STRING ON FILE "KRYWEN".
320  FOR I=1 TO 4
330     PRINT #1,I;A$(I)
340  NEXT I
350  ! *** LINK KRYWEN ON AT END OF SUBPROGRAM.
360  LINK "KRYWEN",5010,400
370  !
380  !
390  ! *** INPUT AND PRINT PARAMETERS FOR SUBPROGRAM.
400  INPUT "LOWER BOUND?",Low
410  PRINT LIN(2),"LOWER BOUND=";Low
420  INPUT "UPPER BOUND?",Up
430  PRINT "UPPER BOUND=";Up
440  ! *** SHOULD PARTIAL RESULTS BE PRINTED?
450  LINPUT "PRINT INTERMEDIATE DATA POINTS (Y/N)?",C$
460  IF (C$="Y") OR (C$="y") OR (C$="N") OR (C$="n") THEN 480
470  GOTO 450
```

```
480  IF (C$="Y") OR (C$="y") THEN Flg=1
490  IF (C$="N") OR (C$="n") THEN Flg=-1
500  ! *** HOW MANY TIMES SHOULD INTERVAL BE HALVED?
510  INPUT "MAX # OF INTERVAL HALVINGS?",Itmax
520  PRINT "MAX # OF INTERVAL HALVINGS=";Itmax
530  ! *** ERROR TOLERANCE BETWEEN SUCCESSIVE CALCULATIONS OF INTEGRAL
.
540  INPUT "ERROR TOLERANCE?",Tol
550  PRINT "ERROR TOLERANCE=";Tol,LIN(1)
560  !
570  !
580  ! *** ALL PARAMETERS HAVE BEEN ESTABLISHED.  CALL Simp.
590  CALL Simp(Low,Up,Int,Flg,Itmax,Tol)
600  !
610  !
620  ! *** PRINT INTEGRAL.
630  PRINT USING 640;Int
640  IMAGE //,"INTEGRAL=",MZ.6DE,5/
650  END
```

HEWLETT hp PACKARD

# SIMULTANEOUS LINEAR EQUATIONS

Subprogram Name: Simultaneous

This subprogram allows the user to solve a set of simultaneous linear equations. If the system has no solution, the user will be advised as such if he checks the determinant of the inverted coefficient matrix immediately after calling the subprogram. If the value of DET is zero, then the system has no (unique) solution.

Subprogram Utilization:

File Name: SIMULS

Calling Syntax: CALL Simultaneous(Coeffs(*), B(*), Noeqs, Solution(*))

Input Parameters:

Coeffs(*) - This is a two-dimensional array holding the coefficients of the system of equations.

B(*) - This is a one-dimensional array holding the right hand sides of the system of equations.

Noeqs - This is the number of equations in the system. Noeqs also determines the dimension of Coeffs(*), B(*), and Solution(*).

Output Parameters:

Solution(*) - This is the solution vector of the system of equations.

Note: All three of the above arrays (Coeffs(*), B(*), Solution(*)) must be dimensioned in the calling program.

Local Variables:

Temparray(*) - This two-dimensional array is allocated by the subprogram to hold the inverse of the matrix Coeffs(*).

Special Considerations and Programming Hints:

1. After calling this subprogram, the user's main program should always check the value of the determinant of the inverse of the coefficient matrix. Since the subprogram "Simultaneous" has already computed the inverse of the coefficient matrix, the value of the determinant can be found by using the system function DET.
2. System Configuration:
   Standard Memory Option
   (Optional) Printer

## Annotated Listing of Subprogram Simultaneous

```
1310 SUB Simultaneous(Coeffs(*),B(*),Noeqs,Solution(*))
1320 OPTION BASE 1
1330 DIM Temparray(Noeqs,Noeqs)          ! Allocate array to hold inverse
1340 MAT Temparray=INV(Coeffs)           ! Compute inverse of coefficient
 matrix
1350 MAT Solution=Temparray*B            ! Multiply inverse with right ha
nd side
1360                                     !  to get the solutioin vector
1370 SUBEXIT
```

Methods and Formulae:

given the system of equations:

$$A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + \ldots + A_{1n}x_n = B_1$$

$$A_{21}x_1 + A_{22}x_2 + A_{23}x_3 + \ldots + A_{2n}x_n = B_2$$

$$A_{31}x_1 + A_{32}x_2 + A_{33}x_3 + \ldots + A_{3n}x_n = B_3$$

$$\vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \qquad \vdots$$

$$A_{n1}x_1 + A_{n2}x_2 + A_{n3}x_3 + \ldots + A_{nn}Y_n = B_n$$

let it be represented by the notation

$$\vec{A}\,x = \vec{b}$$

then

$$A^{-1}\,A\,\vec{x} = A^{-1}\vec{b}$$

and

$$\vec{x} = A^{-1}\,\vec{b}$$

(Provided A is non-singular. If A is singular, then the system has no (unique) solution.)

Because of machine limitations in precision, the inverse of A will not always be exact, thus insuring that the computed solution x will not be exact. The absolute error incurred by round-off error when inverting the matrix is:

$$A\,\vec{x} - \vec{b}$$

REFERENCES:

1. Ralston, Anthony, A First Course In Numerical Analysis (McGray-Hill, 1965) pp.396-397.

Driver Utilization:

File Name: SIMULD

This driver allows the user to enter a system of simultaneous linear equations. The driver then calls the subprogram "Simultaneous" to solve the system. Before the solution is printed, the value of DET is checked. If the value of |DET| is less than 1 E-12, then the driver issues a message saying that the system has no solution. The cutoff point 1 E-12 is somewhat arbitrary, as many non-singular matrices can have determinants that large. The only trouble is, many singular matrices might (because of roundoff error) leave a greater than 1 E-12 in the DET function. The user is advised to use some discretion.

In addition to the computed solution, the driver also prints a residual vector, given by

$$|A\ \vec{x}_c - \vec{b}|$$

(See the cited reference and the Method and Formulae section.)

User Instructions:

1. Insert the Utility Routines cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SIMULD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
5. When "How many equations are to be solved?" appears in the display area:
   a. Enter: The number of equations to be solved.
   b. Press: CONT

6. Perform this step for each of the equations to be solved:
   a. When "Please enter the coefficient for element # i, j of the system" appears in the display area:
      1) Enter: The jth coefficient of the ith equation.
      2) Press: CONT
      3) Repeat step 6a for each coefficient of the ith equation.
   b. When "Please enter the right hand side of equation # i" appears in the display:
      1) Enter: the right-hand side of the ith equation.
      2) Press: CONT
   c. Repeat step 6 for each of the equations in the system.
7. When "Would you like to make any changes (Y/N)?" appears in the display area:
   a. If you would like to make any changes:
      1) Type: Y
      2) Press: CONT
      3) Go to step 8.

      or

   a. If you would not like to make any changes:
      1) Type: N
      2) Press: CONT
      3) Go to step 13.
8. When "Which equation needs to be changed?" appears in the display area:
   a. Enter: The equation number which needs to be changed.
   b. Press: CONT
9. When "Which side of the equation is in error, the left or the right (L/R)?" appears in the display area:
   a. If you want to change the left-hand side of the equation:
      1) Type: L
      2) Press: CONT
      3) Go to step 10.

      or

   a. If you want to change the right-hand side of the equation:
      1) Type: R

2) Press: CONT

3) Go to step 12.

10. When "Which coefficient is wrong?" appears in the display area:

a. Enter: The subscript of the faulty coefficient.

b. Press: CONT

11. When "Please enter the coefficient for element #i, j of the system" appears in the display area:

a. Enter: The new jth coefficient of the ith equation.

b. Press: CONT

c. Go to step 7.

12. When "Please enter the right-hand side of equation #i" appears in the display area:

a. Enter: The new right-hand side of the ith equation.

b. Press: CONT

c. Go to step 7.

13. The complete system of equations will be printed, followed by the solution (if it exists) and an absolute error vector.

14. When "Do you want to run the program again (Y/N)?" appears in the display area:

a. If you want to run the program again:

1) Type: Y

2) Press: CONT

3) Go to step 15.

or

a. If you do not want to run the program again:

1) Type: N

2) Press: CONT

3) The program stops.

15. When "Do you want to modify the previous system or enter a new system (P/N)?" appears in the display area:

a. If you want to modify the previous system:

1) Type: P

2) Press: CONT

3) Go to step 7.

or

a. If you want to start over with a new system:

1) Type: N
2) Press: CONT
3) Go to step 5.

## EXAMPLE

```
Here is the system of equations:
Equation #1:    1*X(1) + 5*X(2) + 4*X(3) + 3*X(4) + 8*X(5) = 6
Equation #2:    7*X(1) + 8*X(2) + 5*X(3) + 4*X(4) + 8*X(5) = 9
Equation #3:    2*X(1) + 0*X(2) + 4*X(3) + 7*X(4) + 3*X(5) = 6
Equation #4:    7*X(1) + 4*X(2) + 8*X(3) + 5*X(4) + 1*X(5) = 3
Equation #5:    1*X(1) + 2*X(2) + 2*X(3) + 4*X(4) + 0*X(5) = 0


Here is the solution:                   Here is the absolute error:
     X(1) =  .866050808319                      .00000000008
     X(2) =  -.717090069282                     .00000000007
     X(3) =  -.374133949209                     .00000000005
     X(4) =  .329099307163                      .00000000009
     X(5) =  1.15357967567                      .00000000001
```

# Driver Listing

```
10   LINK "SIMULS",1320
20   PRINTER IS 16
30   PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONTINUE without enter
ing"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90   Select=16
100  INPUT "Printer select code?",Select
110  IF Select<0 THEN 90
120  PRINT PAGE;
130  PRINT "This program allows the user to input a system of simultan
eous"
140  PRINT "linear equations and solves them.  Below, you are asked to
 enter"
150  PRINT "the number of equations you have to solve (the number of e
quations"
160  PRINT "to be solved is equivalent to the number of independent va
riables)."
170  INPUT "How many equations are to be solved?",Noeqs
180  Noeqs=INT(Noeqs)
190  IF Noeqs=0 THEN Done
200  CALL Simdriver(Noeqs,Select,Repeat)
210  IF NOT Repeat THEN Done
220  GOTO 120
230 Done:   BEEP
240  PRINTER IS 16
250  DISP "PROGRAM COMPLETED"
260  END
270  SUB Simdriver(Noeqs,Select,Repeat)
280  OPTION BASE 1
290  DIM Coeffs(Noeqs,Noeqs),B(Noeqs),Solution(Noeqs),Absoluteerror(No
eqs)
300  PRINT PAGE;"Now you are asked to enter the coefficient matrix of
the system"
310  PRINT "of equations.  The subscript conventions used by the promp
ts issued by this"
320  PRINT "program follow this form:",LIN(1)
330  PRINT "A(1,1)*X(1) + A(1,2)*X(2) + A(1,3)*X(3) +.....+ A(1,n)*X(n
) = B(1)"
340  PRINT "A(2,1)*X(1) + A(2,2)*X(2) + A(2,3)*X(3) +.....+ A(2,n)*X(n
```

```
) = B(2)"
350  PRINT "A(3,1)*X(1) + A(3,2)*X(2) + A(3,3)*X(3) +.....+ A(3,n)*X(n
) = B(3)"
360  IMAGE "      :              :              :                    :
     :"
370  FOR I=1 TO 3
380  PRINT USING 360
390  NEXT I
400  PRINT "A(n,1)*X(1) + A(n,2)*X(2) + A(n,3)*X(3) +.....+ A(n,n)*X(n
) = B(n)",LIN(2)
410  PRINTER IS Select
420  FOR I=1 TO Noeqs
430     PRINT "Equation #"&VAL$(I)&":    ";
440     FOR J=1 TO Noeqs
450        DISP "Please enter the coefficient for element #"&VAL$(I)&"
,"&VAL$(J)&" of the system";
460        INPUT "",Coeffs(I,J)
470        PRINT VAL$(Coeffs(I,J))&"*X("&VAL$(J)&") + ";
480     NEXT J
490     DISP "Please enter the right hand side of equation #"&VAL$(I);
500     INPUT "",B(I)
510     PRINT= "&VAL$(B(I))
520  NEXT I
530  PRINT LIN(2)
540  A$=""
550  INPUT "Would you like to make any changes (Y/N)?",A$
560  IF UPC$(A$)="Y" THEN Change
570  IF UPC$(A$)="N" THEN Run
580  BEEP
590  GOTO 540
600 Change:    INPUT "Which equation needs to be changed?",Whicheq
610            Whicheq=INT(Whicheq)
620            IF (Whicheq>=1) AND (Whicheq<=Noeqs) THEN Fix
630            BEEP
640            GOTO Change
650 Fix:       A$=""
660            INPUT "Which side of the equation is in error, the left
or the right (L/R)?",A$
670            IF UPC$(A$)="L" THEN Left
680            IF UPC$(A$)="R" THEN Right
690            BEEP
700            GOTO 660
710 Left:      J=0
720            INPUT "Which coefficient is wrong?",J
730            J=INT(J)
740            IF (J>=1) AND (J<=Noeqs) THEN Getnew
750            BEEP
```

```
760             GOTO Left
770 Getnew:     DISP "Please enter the coefficient for element #"&VAL$(W
hicheq)&","&VAL$(J)&" of the system";
780             INPUT "",Coeffs(Whicheq,J)
790 Printeq:    PRINT "Equation #"&VAL$(Whicheq)&":      ";
800             FOR J=1 TO Noeqs
810             PRINT VAL$(Coeffs(Whicheq,J))&"*X("&VAL$(J)&") + ";
820             NEXT J
830             PRINT= "&VAL$(B(Whicheq))
840             GOTO 540
850 Right:      DISP "Please enter the right hand side of equation #"&VA
L$(Whicheq);
860             INPUT "",B(Whicheq)
870             GOTO Printeq
880 Run:        GOSUB Printsystem
890             CALL Simultaneous(Coeffs(*),B(*),Noeqs,Solution(*))
900             IF ABS(DET)>1E-12 THEN 940
910             BEEP
920             PRINT "SYSTEM HAS NO SOLUTION"
930             GOTO 980
940             MAT Absoluteerror=Coeffs*Solution
950             MAT Absoluteerror=B-Absoluteerror
960             GOSUB Printsolution
970             BEEP
980             INPUT "Do you want to run the program again (Y/N)?",A$
990             IF UPC$(A$)="N" THEN 1130
1000            IF UPC$(A$)="Y" THEN 1030
1010            BEEP
1020            GOTO 980
1030            INPUT "Do you want to modify the previous system or ente
r a new system (P/N)?",A$
1040            IF UPC$(A$)="P" THEN 1080
1050            IF UPC$(A$)="N" THEN 1110
1060            BEEP
1070            GOTO 1030
1080            PRINT LIN(2)
1090            GOSUB Printsystem
1100            GOTO 540
1110            Repeat=1
1120            SUBEXIT
1130            Repeat=0
1140            SUBEXIT
1150 Printsystem:!
1160            PRINT LIN(2);"Here is the system of equations:"
1170            FOR I=1 TO Noeqs
1180            PRINT "Equation #"&VAL$(I)&":    ";
1190            FOR J=1 TO Noeqs
```

```
1200           PRINT VAL$(Coeffs(I,J))&"*X("&VAL$(J)&") + ";
1210           NEXT J
1220           PRINT= "&VAL$(B(I))
1230           NEXT I
1240           PRINT LIN(2)
1250           RETURN
1260 Printsolution:  !
1270           PRINT "Here is the solution:                Here is the a
bsolute error:"
1280           FOR I=1 TO Noeqs
1290           PRINT SPA(5);"X("&VAL$(I)&") = "&VAL$(Solution(I));TAB(4
5);VAL$(Absoluteerror(I))
1300           NEXT I
1310           RETURN
```

This section of programs is a general purpose Information Management System. There are three basic divisions in these programs:

1) Backup program
2) Initialization program
3) Access programs (all the rest of the programs)

The Backup program allows the user to copy all of the programs from one mass storage medium to another.

The Initialization program initializes a data file according to the user's specifications for use with this system of programs. The user may use any one of four different types of fields in any number and any order he chooses:

1) full-precision numeric field,
2) string field (of any length),
3) full-precision numeric array (of any number of dimensions up to 6, and any range of subscripts on each of the dimensions), and
4) string array (the dimensions are limited only by the same conditions as those that govern numeric arrays).

Each of these fields may have a title associated with it (the title may be up to 30 characters long).

One of the fields defined by the user (mentioned above) is selected (by the user) as a key field. The key of a record is its identifier. In order to avoid confusion with other records, the key value must be unique to its record--no other record may have the same key value (or identifier).

The Access programs allow the user to interact with his data base. Once the user has set up his data file, he may enter data into it, edit data that is already there, delete obsolete records, and so on. The user may have reports printed out, and he has the option of formatting the report himself by means of

a question and answer session with the 9845 , or he may let the 9845 format a default report. The user may select any field in his data record to sort on for reports.

This system will run with the standard memory option, however, performance will be better with additional memory. This is because some subroutines must be linked in as they are needed with the standard memory whereas additional memory allows all the routines to reside in memory at once.

The user should make note of the fact that this system was not designed to run using the tape cartridge as a storage medium. Due to the Unified Mass Storage command structure of the 9845A, the system will run using tape cartridges, but because of the high amount of access to the storage device, a tape would probably give unsatisfactory performance. A disk drive (like the 9885 or the 7900 disk family) is highly recommended for this system.

HEWLETT hp PACKARD

# BACKUP OF INFORMATION MANAGEMENT SYSTEM PROGRAMS

This program allows you to copy all of the programs needed for the Information Management System from one mass storage device to another. This is to allow you to transfer the programs from the tape cartridge on which they are originally stored to another storage medium. There are two reasons that you might want to copy the programs to another device. One reason is for room. The Utility Library cartridge 1 is full enough that you would not be able to get a data base of any significant size on it. The second reason is speed. The Information Management System calls for extensive linking between programs. A tape cartridge is very slow in this respect. It would be much faster to use a disk device of some sort (i.e., the 9885M floppy drive or the 7900 hard disk family) to store the programs on.

Program Utilization:

File Name: DBBCKP

Variables:

A$(*) - This string array holds the names of the programs in the Information Management System which are to be copied.

C - Return variable which sees whether or not a file already exists

I - Used as a loop counter

S$ - Holds the device code of the source

T$ - Holds the device code of the target of the program

Special Considerations and Programming Hints:

1. The User Instructions for this program assume that you are copying from the primary tape transport. Actually the program may be used to copy the Information Management System from any device. The only step that needs to be changed in the user instructions to generalize the process is step #4. Here, instead of typing T15, you would type the device code of whatever device you were copying from.

2. While copying the programs to the new device, the backup program has the CHECK READ feature of the 9845 on to insure that the programs are copied correctly.

User Instructions:

1. Insert the Utility Routines Cartridge 1 into the primary transport (i.e., the transport above the special function keys). Make sure the storage medium you are copying the programs to is initialized (refer to the Operating and Programming Manual for instructions on initializing a mass storage medium).
2. Load the file:
   a. Type: GET "DBBCKP: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Which mass storage device presently has the programs?" appears in the display area of the CRT:
   a. Type: T15
   b. Press: CONT
5. When "Which mass storage device do you want to copy the programs to?" appears in the display area:
   a. Type: The mass storage identifier of the device to which you wish to copy the programs (examples: To copy the programs to a floppy disk on select code 8, you would type F8. To copy the programs to the secondary tape transport above the TYPING FUNCTION keys, you would type T14. To copy the programs to a 7905 disk drive on select code 12, you would type Y12.)
   b. Press: CONT
6. The programs' names will be listed on the CRT, and the program currently being copied will be displayed in the display area of the CRT, so you can tell how far the backup process has gone. After all the programs have been copied over, the message "BACKUP COMPLETE" will be displayed, and a beep will sound. At this point, the program is finished.

# Listing of Backup Program

```
10   PRINTER IS 16
20   CHECK READ
30   PRINT PAGE
40   DATA DBINIT,DBACCS,DBDRVR,DBREPO,DBSUB1,DBEXSS,DBEXSN,DBLTKY,DBBC
KP
50   DIM A$(1:9)[6]
60   READ A$(*)      ! Initialize the file array
70   FOR I=1 TO 9
80   PRINT A$(I)
90   NEXT I
100  S$[1]=T$[1]=":"
110  INPUT "Which mass storage device presently has the programs?",S$[
2]
120  INPUT "Which mass storage device do you want to copy the programs
 to?",T$[2]
130  FOR I=1 TO 9
140  DISP A$(I)
150  ASSIGN #1 TO A$(I)&T$,C     !  Check to see if the file already ex
ists.
160  IF C=1 THEN 180
170  PURGE A$(I)&T$              !  If the file already exists, purge i
t.
180  COPY A$(I)&S$ TO A$(I)&T$
190  NEXT I
200  BEEP
210  DISP "BACKUP COMPLETE"      !  All done
220  END
```

HEWLETT hp PACKARD

# DATA FILE INITIALIZATION PROGRAM

This program initializes a data file according to the user's specifications. The program asks the user what kind of fields he wants in the data file, what he wants to title each field, and which field to use as a key. (The key field is the field that identifies each data record as being unique from every other data record.) Using the above information, the program computes how long a defined record must be to hold all of the specified fields. Then it computes how many of these defined records are needed to store the necessary overhead information, and asks the user for the maximum number of data records he wants the file to hold. The file is created, the overhead information is stored, and the free space is initialized to a linked list for easy maintenance. Finally, a directory file is built to keep track of the used records in the data file.

Note: The user must run this program for every data file he wishes to establish, because the access programs scan the data files for information relating to the format of the file. That information is stored away in the data file by this program.

Program Utilization:

File Name: DBINIT

Variables:

A - The record number in the data file where the titles of the fields start

A$ - Used to hold the user's replies to Y/N questions

Answer$ - Used to hold the user's replies to Y/N questions

B - The record number in the data file where the PRINT # statements used to print the data records to the data file start

Bytesperrecord - The number of bytes needed to store one of the user's data records (i.e., the defined record size of the data file)

C - The record number in the data file where the READ # statements used to read the data records from the data file start

Check - This variable is used in conjunction with ASSIGN statements to determine whether or not a data file already exists

Count - Used to count the data fields

D - The record number in the data file where the DIM statements(s) needed to allocate the arrays in the user's records start

Dimensionality - Used to hold the dimensionality of the user's array

Dimhead - The amount of overhead (in bytes) needed to store the DIM statements(s)

Dimrec - The number of defined records needed to store the DIM statements(s)

Dimstat$ - This variable holds the DIM statement until it grows too large. Then it is saved away and re-initialized.

Dirnorecs - The number of defined records needed for the directory file record

Dirreclength - The defined record length of each directory file record

E - The record number in the data file where the key assignment statement is kept

F - The record number in the data file where the name of the directory file is kept

Filename$ - Used to hold file names that the user enters

Forcount - Counts the number of format codes that are needed

Formats_per_rec - The number of format codes that can be stored in one of the user's data records

Forrec - The number of data records needed to store all the format codes

G - The record number in the data file where the head of the free list starts. Also, the total number of data records needed to store the overhead information.

High - The upper subscript of a dimension

I - Loop counter

J - Loop counter

K - Loop counter

Keyhead - The amount of overhead (in bytes) needed to store the key assignment

Keyl - The number of bytes the key (whether string or numeric) will take up when stored on mass memory (used in computing Dirreclength)

Keylength - The amount of storage (in bytes) a key takes up when it is stored internally

Keyrec - The number of defined records in the data file needed to store the key assignment statement

Keysinmemory - The maximum number of keys (with their associated record pointers) which will fit into less than 2K bytes of memory

Keystat$ - This string is used to hold the key assignment statement

Keytitle$ - The title of the field the user selects for a key

Keytype - This variable is used to distinguish between the two possible types of keys the user may select: 1) A numeric key or 2) a string key

Klength - Used for temporary storage of Keylength

Length - Whenever the user selects a string field, Length holds the length of the string field (used in constructing DIM statements)

Low - The lower subscript of a dimension

Maxsize - The maximum number of data records which the user needs

Numarrays - The number of numeric arrays in the user's data record

Numeric - The number of simple numeric fields in the user's data record

Poskey - The position of the key field with respect to the other fields of the data record

Prthead - The amount of overhead (in bytes) needed to store the PRINT # statement(s)

Prtrec - The number of defined records needed to store the PRINT # statement(s)

Prtstat$ - This string holds each PRINT # statement until it becomes too long, at which point it is stored away and re-initialized

Readhead - The amount of storage (in bytes) needed to store the READ # statement(s)

Readrec - The number of data records needed to store the READ # statement(s)

Readstat$ - This string holds each READ # statement until it becomes too long, at which point it is stored away and re-initialized

Recover - Used to allow the user to recover from asking for

too many records in his data file

S - Used for temporary storage of lower subscripts

Scratch$(*) - This array holds the file names of the scratch files which are used for temporary storage of the overhead information

Strinarrays - Counts the number of string arrays in the user's data record

Strings - Counts the number of simple string fields in the user's data record

T - Used for temporary storage of upper subscripts

Title$ - This string holds the titles of the fields of the data records.

Titlehead - The amount of overhead needed (in bytes) to store the titles of the fields in the data record

Titlerec - The number of defined records needed to store all the titles

Type - This variable is the user's selection of the next type of field in the data record (1 for numeric, 2 for string, 3 for numeric array, and 4 for string array)

Varindex - The variable index of the selected key field

Varnam$ - This string holds a constructed variable name which in turn is used in the construction of the PRINT # statement(s), the READ # statement(s), the DIM statement(s), and the key assignment statement

X - Temporary variable for format codes

Y - Temporary variable for format codes

Z - Temporary variable for format codes

Special Considerations and Programming Hints:

1. It should be noted that the user does not have to keep his programs on the same mass storage device as his data. This and later programs allow the user to have his programs on one mass storage device, and his data on another mass storage device.

2. The user should be careful when selecting a key field, to select one which will not call for two or more records having the same value in the key field. A field having peoples' names in it is a notoriously bad example of a key field, unless you are certain that no two people in your data base are going to have the same name. If you try to use a certain key value more than once in your data base, the data entry program will detect it and make you enter a different value for the key. So take care when assigning the key field of the data file. (See step #26 in the user instructions.)

User Instructions:

1. Insert the storage medium containing the Information Management Routines into the mass storage device you wish to use for the programs (see the section titled Special Considerations and Programming Hints).
2. Load the file:
   a. Type: LOAD "DBINIT:(Mass Storage Unit Specifier)" where (Mass Storage Unit Specifier) defines which device you are loading the programs from (examples: To load from the primary tape transport, type LOAD "DBINIT:T15". To load from a floppy disk on select code 8, type LOAD "DBINIT:F8".
   b. Press: CONT
3. Start the program:
   a. Press: RUN
4. When "Which mass storage device are you using?" appears in the display area of the CRT:

a. Type: (Mass Storage Unit Specifier) where (Mass Storage Unit Specifier) is of the same format as that in step 2, and which identifies that mass storage device being used to store your data. (Note: The data need not be stored on the same device as the programs--see #1 under the section titled Special Considerations and Programming Hints.)

b. Press: CONT

5. Wait until the program is finished setting up some scratch files. Then proceed to step 6.

6. When "Please enter the data type of the next field of the record" appears in the display area:

   a. Enter: A number from 0 through 4, depending upon which data type the next field of your record will be (the types are listed on the CRT)

   b. Press: CONT

7. Depending upon which number was entered in part 6a, go to one of the following steps:

   a. If you entered a 0 (end of record), go to step 26.
   b. If you entered a 1 (numeric field), go to step 8.
   c. If you entered a 2 (string field), go to step 11.
   d. If you entered a 3 (numeric array), go to step 15.
   e. If you entered a 4 (string array), go to step 19.
   f. If you entered anything else, go back to step 6a.

8. When "Numeric field (Y/N)?" appears in the display area of the CRT:

   a. If you want a numeric field to be the next field in the data record:

      1) Type: Y
      2) Press: CONT
      3) Go to step 8b.

      or

   a. If you made a mistake when entering the type of the next field and you do not want the next field to be a numeric field:

1) Type: N
2) Press: CONT
3) Go to step 6.

b. When "Please enter the title" appears in the display area:
1) Type: The title you want to use for this field (limited to 30 characters)
2) Press: CONT

9. When "Do you want to make any changes in the title (Y/N)?" appears in the display area:
a. If you want to change the title:
1) Type: Y
2) Press: CONT
3) Go to step 10.
or
a. If you do not want to change the title:
1) Type: N
2) Press: CONT
3) Go to step 6.

10. When "Make any changes you want and press CONT" appears in the display area:
a. Make any desired corrections in the title.
b. Press: CONT
c. Go to step 9.

11. When "String field (Y/N)?" appears in the display area:
a. If you want a string field to be the next field in the data record:
1) Type: Y
2) Press: CONT
3) Go to step 11b.
or
a. If you made a mistake when entering the type of the next field and you do not want the next field to be a string field.
1) Type: N
2) Press: CONT
3) Go to step 6.

b. When "Please enter the title" appears in the display area:

1) Type: The title you want to use for this field (limited to 30 characters)

2) Press: CONT

12. When "Do you want to make any changes in the title (Y/N)?" appears in the display area:

a. If you want to change the title:

1) Type: Y

2) Press: CONT

3) Go to step 12b.

or

a. If you do not want to change the title:

1) Type: N

2) Press: CONT

3) Go to step 13.

b. When "Make any changes you want and press CONT" appears in the display area:

1) Make any desired corrections in the title.

2) Press: CONT

3) Go to step 12.

13. When "Please enter the string length" appears in the display area:

a. Enter: The maximum string length you will need in this field.

b. Press: CONT

14. Go to step 6.

15. When "Numeric array (Y/N)?" appears in the display area:

a. If you want a numeric array to be the next field in the data record:

1) Type: Y

2) Press: CONT

3) Go to step 15b.

or

a. If you made a mistake when entering the type of the next field and you do not want the next field to be a numeric array:

1) Type: N
2) Press: CONT
3) Go to step 6.

b. When "Please enter the title" appears in the display area:
1) Type: The title you want to use for this field (limited to 30 characters)
2) Press: CONT

16. When "Do you want to make any changes in the title (Y/N)?" appears in the display area:
a. If you want to change the title:
1) Type: Y
2) Press: CONT
3) Go to step 16b.
or
a. If you do not want to change the title:
1) Type: N
2) Press: CONT
3) Go to step 17.
b. When "Make any changes you want and press CONT" appears in the display area:
1) Make any desired corrections in the title.
2) Press: CONT
3) Go to step 16.

17. When "Please enter the number of dimensions in your array" appears in the display area:
a. Enter: The number of dimensions in your array (limited to 6)
b. Press: CONT

18. When "Is [X] okay (Y/N)?" appears in the display (where [X] is the number entered in part 17a):
a. If the correct number was entered in step 17a:
1) Type: Y
2) Press: CONT
3) Go to step 24.
or
a. If you entered incorrectly in step 17a:

1) Type: N
2) Press: CONT
3) Go to step 17.

19. When "String array (Y/N)?" appears in the display area:

a. If you want a string array to be the next field in the data record:

1) Type: Y
2) Press: CONT
3) Go to step 19b.

or

a. If you made a mistake when entering the type of the next field and you do not want the next field to be a string array:

1) Type: N
2) Press: CONT
3) Go to step 6.

b. When "Please enter the title" appears in the display area:

1) Type: The title you want to use for this field (limited to 30 characters)
2) Press: CONT

20. When "Do you want to make any changes in the title (Y/N)?" appears in the display area:

a. If you want to change the title:

1) Type: Y
2) Press: CONT
3) Go to step 20b.

or

a. If you do not want to change the title:

1) Type: N
2) Press: CONT
3) Go to step 21.

b. When "Make any changes you want and press CONT" appears in the display area:

1) Make any desired corrections in the title.
2) Press: CONT

3) Go to step 20.

21. When "Please enter the string length of each element of the array" appears in the display area of the CRT:
    a. Enter: The string length of each element of the string array
    b. Press: CONT
22. When "Please enter the number of dimensions in your string array" appears in the display area:
    a. Enter: The number of dimensions in the string array (limited to 6)
    b. Press: CONT
23. When "Is [X] okay (Y/N)?" appears in the display area:
    a. If you entered the correct number in part 22a:
        1) Type: Y
        2) Press: CONT
        3) Go to step 24.

        or

    a. If you entered incorrectly in part 22a:
        1) Type: N
        2) Press: CONT
        3) Go to step 22.
24. Do this step [X] times, where [X] is the number of dimensions in the array.
    a. When "Dimension # i lower subscript?" appears in the display:
        1) Enter: The lower subscript of the ith dimension
        2) Press: CONT
    b. When "Dimension # i upper subscript?" appears in the display:
        1) Enter: The upper subscript of the ith dimension
        2) Press: CONT
    c. When "Lower dimension: l Upper dimension: u. Okay (Y/N)?" appears in the display area:
        1) If the displayed dimensions are correct:
            a) Type: Y
            b) Press: CONT
            c) Go to step 24d.

or

1) If the displayed dimensions are incorrect:
   a) Type: N
   b) Press: CONT
   c) Go to step 24a.

d. Repeat step 24 as often as necessary. (Note: If a beep is sounded and the message "The lower subscript can not be greater than the upper subscript" is displayed, you will have to re-enter both the lower and upper subscript of the ith dimension.)

25. When the subscript ranges for each of the dimensions of the array have been entered, go to step 6.

26. When "End of record (Y/N)?" appears in the display area:

a. If this is the end of the record:
   1) Type: Y
   2) Press: CONT
   3) Go to step 27.

   or

a. If you made a mistake when entering the type of the next field and you do not want this to be the end of the record:
   1) Type: N
   2) Press: CONT
   3) Go to step 6.

27. When the titles of all the fields you have entered are printed on the screen, and "Please enter the title of the field you wish to use for a key" appears in the display area of the CRT:

a. Type: The title of the field you wish to use as a key

b. Press: CONT

c. 1) If you type in the title of a non-existent field (i.e., if you misspell the title) go back to step 27.
   2) If the field you have designated for the key is a numeric or string field, go to step 29.
   3) If the field you have designated for the key is a numeric or string array, go to step 28.

28. Perform this step for each of the dimensions in the array.
    a. When "Please enter subscript #j for the key" appears in the display area:
        1) Enter: The jth subscript of the element of the arr being used as a key
        2) Press: CONT
        3) If you enter a subscript that does not fall within the valid range of subscripts for the jth dimension in step 27a1), a message will be displayed to that effect, and the valid subscript range of the jth di mension will be displayed also.
    b. Repeat step 28 as often as necessary.
29. When "What do you want to name your file?" appears in the di play area of the CRT:
    a. Type: A valid file name (A file name may be any combin tion of characters up to six characters long).
    b. Press: CONT
    c. Note that the file name must not already exist. If a file having the name entered in step 29a already exists you will be asked to enter a different file name. The program will not purge the file that already exists, because it might contain material that is valuable to the user. If you decide that the already existing file does not contain anything you need, you may purge the file by typing PURGE "(file name)" and pressing EXECUTE Then you may re-enter the file name in response to the question being asked by the program.
30. When "What is the upper bound on the number of records you need?" appears in the display:
    a. Enter: The maximum number of data records you will nee to store in your file
    b. Press: CONT
    c. If you ask for more records than your storage medium ha room for, you will be informed that there is not enough room, and you will be given two choices of how to recov from this mishap. 1) You may ask for a lower number o

records in your data file, or 2) you may use a different storage medium (get a new tape or floppy diskette) and re-run the program.

31. At this point the program will set up your file. Various messages will be printed on the CRT informing you what the program is doing. This is merely to keep you informed. You will be told when a response is expected.

32. When "Please enter the name of the directory" appears in the display area of the CRT:
    a. Type: A valid file name
    b. Press: CONT
    c. Note that this file name should be different from the one entered in step 29. Refer to the note in 29c for detail concerning what happens if you enter the name of an already existing file.

33. Wait until the message "All done initializing" appears in the display area of the CRT and the run light goes off. Then the program is finished and your data file is ready for use.

# Listing of File Initialization Program

```
10    OPTION BASE 1
20    SERIAL
30    DIM Vernum$[160],Prtstat$[160],Readstat$[160],Dimstat$[160],Keyst
at$[160]
40    DIM Filename$[6],A$[3],Scratch$(6)[6],Title$[30],Keytitle$[30]
50    Numeric=Strings=Numarrays=Strinarrays=0
60    Dimhead=Prthead=Readhead=Titlehead=4
70    PRINTER IS 16
80    PRINT PAGE;"Please enter the device code and select code of the m
ass"
90    PRINT "storage medium you are using.  For example, if you have a
9885M on"
100   PRINT "select code 8, you would enter F8.  If you are using the p
rimary"
110   PRINT "tape transport, you would enter T15. If you are using a 79
05 disk"
120   PRINT "unit on select code 12, you would enter Y12."
130   LINPUT "Which mass storage device are you using?",A$
140   ON ERROR GOTO Bomb1
150   MASS STORAGE IS ":"&A$
160   REWIND ":"&A$
170   OFF ERROR
180   GOTO 230
190 Bomb1:  BEEP
200   DISP "ILLEGAL.  TRY AGAIN."
210   WAIT 1000
220   GOTO 130
230   PRINT PAGE;"There are several scratch files needed for this progr
am to run"
240   PRINT "properly.  Right now, the program is setting them up.  Ple
ase wait"
250   PRINT "until everything is ready."
260   DATA PRT@@@,REA@@@,DIM@@@,FOR@@@,TTL@@@,KEY@@@
270   READ Scratch$(*)
280   FOR I=1 TO 6                        ! Initialize the scratch files
290      ASSIGN #I TO Scratch$(I),Check
300      IF Check=0 THEN PURGE Scratch$(I)
310      CREATE Scratch$(I),20
320      ASSIGN #I TO Scratch$(I)
330      BUFFER #I
340   NEXT I
350   BEEP


********************
```

```
70  Prtstat$="PRINT#9;"              ! Initialize the PRINT#, READ#, &
IM
80  Readstat$="READ#9;"              !  statements
90  Dimstat$="DIM "


******************


10 Instruct1:  PRINT PAGE;"Here are the valid data types:"
20       PRINT SPA(5),"0->End of Record",LIN(1),SPA(5),"1->Numeric Fi
ld"
30       PRINT SPA(5),"2->String Field",LIN(1),SPA(5),"3->Numeric Arr
y"
40       PRINT SPA(5),"4->String Array",LIN(1)
50 Entertype: Type=0
60       BEEP
70       INPUT "Please enter the data type of the next field of the r
cord",Type
80       IF (Type<0) OR (Type>=5) THEN Entertype
90       Type=INT(Type)
00       ON Type+1 GOTO Endrecord,Numerictype,Stringtype,Anumtype,Ast
ype


******************


20 Numerictype: A$=""
30       INPUT "Numeric field (Y/N)?",A$
40       IF UPC$(A$)="Y" THEN 580
50       IF UPC$(A$)="N" THEN Entertype
60       BEEP
70       GOTO Numerictype
80       Numeric=Numeric+1  ! This section allows for numeric fields
90       GOSUB Entertitle
00       Varnam$="N"&VAL$(Numeric)&","
10       Prtstat$=Prtstat$&Varnam$
20       Readstat$=Readstat$&Varnam$
30       GOSUB Lenprt
40       GOSUB Lenread
50       PRINT #4;1
60       GOTO Instruct1
```

```
********************


680 Stringtype: A$=""
690         INPUT "String field (Y/N)?",A$
700         IF UPC$(A$)="Y" THEN 740
710         IF UPC$(A$)="N" THEN Entertype
720         BEEP
730         GOTO Stringtype
740         Strings=Strings+1    ! This section takes care of string fiel
ds
750         GOSUB Entertitle
760 Enterlength: Length=PI
770         INPUT "Please enter the string length",Length
780         IF (Length<>PI) AND (Length>=1) THEN 810
790         BEEP
800         GOTO Enterlength
810         Length=INT(Length)
820         IF Length<=200 THEN 870
830         BEEP
840         DISP "SORRY.  MAXIMUM STRING LENGTH IS 200 CHARACTERS"
850         WAIT 1000
860         GOTO Enterlength
870         Varnam$="S"&VAL$(Strings)&"$"
880         Dimstat$=Dimstat$&Varnam$&"["&VAL$(Length)&"],"
890         Prtstat$=Prtstat$&Varnam$&","
900         Readstat$=Readstat$&Varnam$&","
910         GOSUB Lenprt
920         GOSUB Lenread
930         GOSUB Lendim
940         PRINT #4;2,Length
950         GOTO Instruct1


********************


970 Anumtype: A$=""
980         INPUT "Numeric array (Y/N)?",A$
990         IF UPC$(A$)="Y" THEN 1030
1000        IF UPC$(A$)="N" THEN Entertype
1010        BEEP
1020        GOTO Anumtype
1030        Numarrays=Numarrays+1 ! This section handles numeric arrays
1040        GOSUB Entertitle
```

```
050 Enterdim:Dimensionality=PI
060       INPUT "Please enter the number of dimensions in your array",
imensionality
070       IF NOT ((Dimensionality<1) OR (Dimensionality=PI) OR (Dimens
onality>=7)) THEN 1120
080       BEEP
090       DISP "ILLEGAL DIMENSIONS"
100       WAIT 1000
110       GOTO Enterdim
120       Dimensionality=INT(Dimensionality)
130       DISP "Is ";Dimensionality;" okay (Y/N)";
140       Answer$="Y"
150       INPUT Answer$
160       IF (Answer$="Y") OR (Answer$="y") THEN 1190
170       IF (Answer$="N") OR (Answer$="n") THEN Enterdim
180       GOTO 1130
190       Varnam$="Na"&VAL$(Numarrays)
200       Prtstat$=Prtstat$&Varnam$&"(*),"
210       Readstat$=Readstat$&Varnam$&"(*),"
220       Varnam$=Varnam$&"("
230       PRINT #4;3,Dimensionality
240       GOSUB Dimenter
250       Dimstat$=Dimstat$&Varnam$&","
260       GOSUB Lenprt
270       GOSUB Lenread
280       GOSUB Lendim
290       GOTO Instruct1


******************


310 Astrtype:A$=""
320       INPUT "String array (Y/N)?",A$
330       IF UPC$(A$)="Y" THEN 1370
340       IF UPC$(A$)="N" THEN Entertype
350       BEEP
360       GOTO Astrtype
370       Strinarrays=Strinarrays+1  ! This section takes care of stri
g arrays
380       GOSUB Entertitle
390 Enterlen:Length=PI
400       INPUT "Please enter the string length of each element of the
array",Length
410       IF (Length>=1) AND (Length<>PI) THEN 1440
420       BEEP
430       GOTO Enterlen
```

```
1440        Length=INT(Length)
1450        IF Length<=200 THEN Entrdim
1460        BEEP
1470        DISP "SORRY.  MAXIMUM STRING LENGTH IS 200 CHARACTERS"
1480        WAIT 1000
1490        GOTO Enterlen
1500 Entrdim:Dimensionality=PI
1510        INPUT "Please enter the number of dimensions in your string
array",Dimensionality
1520        IF NOT ((Dimensionality<1) OR (Dimensionality>=7) OR (Dimens
ionality=PI)) THEN 1570
1530        BEEP
1540        DISP "ILLEGAL DIMENSIONS"
1550        WAIT 1000
1560        GOTO Entrdim
1570        Dimensionality=INT(Dimensionality)
1580        DISP "Is ";Dimensionality;" okay (Y/N)";
1590        Answer$="Y"
1600        INPUT Answer$
1610        IF (Answer$="y") OR (Answer$="Y") THEN 1650
1620        IF (Answer$="n") OR (Answer$="N") THEN Entrdim
1630        BEEP
1640        GOTO 1580
1650        PRINT #4;4,Length,Dimensionality
1660        Varnam$="Sa"&VAL$(Strinarrays)&"$"
1670        Prtstat$=Prtstat$&Varnam$&"(*),"
1680        Readstat$=Readstat$&Varnam$&"(*),"
1690        Varnam$=Varnam$&"("
1700        GOSUB Dimenter
1710        Varnam$=Varnam$&"["&VAL$(Length)&"],"
1720        Dimstat$=Dimstat$&Varnam$
1730        GOSUB Lenprt
1740        GOSUB Lenread
1750        GOSUB Lendim
1760        GOTO Instruct1


********************


1780 Dimenter: ! This subroutine stores the dimensions of the array
1790           FOR I=1 TO Dimensionality
1800 Reset:    Low=High=PI
1810           DISP "Dimension #";I;"      lower subscript";
1820           INPUT Low
1830           IF Low<>PI THEN 1860
1840           BEEP
```

```
850             GOTO 1810
860             Low=INT(Low)
870             DISP "Dimension #";I;"      upper subscript";
880             INPUT High
890             IF High<>PI THEN 1920
900             BEEP
910             GOTO 1870
920             High=INT(High)
930             IF High>=Low THEN Verify
940             BEEP
950             DISP "The lower subscript can't be greater than the upp
r subscript"
960             WAIT 3000
970             GOTO Reset
980 Verify:     A$=""
990             DISP "Lower dimension: "&VAL$(Low)&"    Upper dimension
 "&VAL$(High)&".  Okay (Y/N)?";
000             INPUT "",A$
010             IF UPC$(A$)="Y" THEN Okay1
020             IF UPC$(A$)="N" THEN Reset
030             BEEP
040             GOTO Verify
050 Okay1:      Varnam$=Varnam$&VAL$(Low)&":"&VAL$(High)&","
060             PRINT #4;Low,High
070             NEXT I
080             Varnam$[LEN(Varnam$)]=")"
090             RETURN


******************


 10 Entertitle: ! This subroutine stores the titles of the fields"
 20             PRINT PAGE;"Please enter the title of the field."
 30             PRINT "If you misspell the title, you will be given the
hance"
 40             PRINT "to correct it.",LIN(2)
 50             LINPUT "Please enter the title",Title$
 60             PRINT LIN(2),"Title: ";Title$
 70             A$=""
 80             INPUT "Do you want to make any changes in the title (Y/
?",A$
 90             IF UPC$(A$)="Y" THEN 2230
 00             IF UPC$(A$)="N" THEN 2250
 10             BEEP
 20             GOTO 2170
 30             EDIT "Make any changes you want and press CONT",Title$
```

```
2240            GOTO 2160
2250            PRINT #5;Title$
2260            Titlehead=Titlehead+4+2*INT((LEN(Title$)+1)/2)
2270            RETURN

********************

2290 Lenprt: ! This subprogram checks to make sure that the PRINT# sta
tement
2300          ! isn't getting too long.
2310            IF LEN(Prtstat$)<130 THEN RETURN
2320            Prtstat$[LEN(Prtstat$)]=""    ! Wipe out the trailing co
mma
2330            PRINT #1;Prtstat$
2340            Prthead=Prthead+2*INT((LEN(Prtstat$)+1)/2)+4
2350            Prtstat$="PRINT#9;"
2360            RETURN
2370 Lenread: ! This subroutine checks to make sure the READ# statemen
t isn't
2380           !  getting too long
2390            IF LEN(Readstat$)<130 THEN RETURN
2400            Readstat$[LEN(Readstat$)]=""  ! Wipe out the trailing c
omma
2410            PRINT #2;Readstat$
2420            Readhead=Readhead+2*INT((LEN(Readstat$)+1)/2)+4
2430            Readstat$="READ#9;"
2440            RETURN

********************

2460 Lendim: ! This subroutine checks to make sure the DIM statement i
sn't
2470          ! getting too long.
2480            IF LEN(Dimstat$)<120 THEN RETURN
2490            Dimstat$[LEN(Dimstat$)]=""  ! Wipe out the trailing com
ma
2500            PRINT #3;Dimstat$
2510            Dimhead=Dimhead+2*INT((LEN(Dimstat$)+1)/2)+4
2520            Dimstat$="DIM "
2530            RETURN
```

```
******************


550 Endrecord: A$=""
560       INPUT "End of record (Y/N)?",A$
570       IF UPC$(A$)="Y" THEN 2610
580       IF UPC$(A$)="N" THEN Entertype
590       BEEP
600       GOTO Endrecord
610       PRINT #4;0
620       PRINT #5;END    ! Mark the end of the Title file
630       !  Store the DIM, READ#, and PRINT# statements
640 Testprt: IF Prtstat$="PRINT#9" THEN Testread
650       Prtstat$[LEN(Prtstat$)]=""  !Wipe out the trailing comma
660       PRINT #1;Prtstat$
670       Prthead=Prthead+2*((LEN(Prtstat$)+1)/2)+4
680 Testread: PRINT #1;END
690       IF Readstat$="READ#9" THEN Testdim
700       Readstat$[LEN(Readstat$)]=""
710       PRINT #2;Readstat$
720       Readhead=Readhead+2*INT((LEN(Readstat$)+1)/2)+4
730 Testdim: PRINT #2;END
740       IF Dimstat$="DIM " THEN Figurelength
750       Dimstat$[LEN(Dimstat$)]=""
760       PRINT #3;Dimstat$
770       Dimhead=Dimhead+2*INT((LEN(Dimstat$)+1)/2)+4


******************


890 Figurelength: ! Compute how many bytes each record will take up
900       ! Read through the record format file "FOR!%@" (#4)
910       READ #4,1       ! Set the serial pointer to the beginning
920       READ #5,1       ! Reset the title file
930       PRINT PAGE,"Here are the fields you have entered:"
940       READ #5;Title$
950       PRINT SPA(5);Title$
960       IF TYP(5)=2 THEN 2840
970       READ #5,1
980       LINPUT "Please enter the title of the field you wish to use f
 a key",Keytitle$
990       Poskey=1
000       READ #5;Title$
010       IF Keytitle$=Title$ THEN 2950
```

```
2920        IF (TYP(5)<>2) AND (TYP(5)<8) THEN 2870
2930        Poskey=Poskey+1
2940        GOTO 2900
2950        Count=1


********************


2970        Forcount=0
2980 Readfor: READ #4;X
2990        Forcount=Forcount+1
3000        IF Count=Poskey THEN GOSUB Keyassignment
3010        Count=Count+1
3020        IF X=0 THEN Endfor
3030        ON X GOSUB Nu,St,Na,Sa
3040        GOTO Readfor


********************


3060 REM         X,Y,Z,S,T, and I are all destroyed here
3070 Nu:         Bytesperrecord=Bytesperrecord+10
3080             RETURN
3090 St:         READ #4;X
3100             Forcount=Forcount+1
3110             Bytesperrecord=Bytesperrecord+4+2*INT((X+1)/2)
3120             RETURN
3130 Na:         READ #4;X
3140             Forcount=Forcount+1
3150             Y=1
3160             FOR I=1 TO X
3170             READ #4;S,T
3180             Forcount=Forcount+2
3190             Y=Y*(T+1-S)
3200             NEXT I
3210             Bytesperrecord=Bytesperrecord+Y*10
3220             RETURN
3230 Sa:         READ #4;X,Y
3240             Forcount=Forcount+2
3250             Z=1
3260             FOR I=1 TO Y
3270             READ #4;S,T
3280             Forcount=Forcount+2
3290             Z=Z*(T+1-S)
3300             NEXT I
```

```
310           Bytesperrecord=Bytesperrecord+Z*(2*INT((X+1)/2)+4)
320           RETURN

******************

340 Keyassignment: Varindex=0
350           READ #4,1
360           FOR I=1 TO Count-1
370                READ #4;Y
380                IF Y=X THEN Varindex=Varindex+1
390                ON Y GOSUB Read1,Read2,Read3,Read4
400           NEXT I
410           READ #4;Y
420           ON X GOSUB Key1,Key2,Key3,Key4
430           RETURN

******************

450 Key1:          Keystat$="Key=N"&VAL$(Varindex+1)&"    !  "&Keytitl

460                Keytype=1
470                PRINT #6;Keystat$
480                Keyhead=2*INT(LEN(Keystat$)+1)/2+4
490                Keylength=8
500                Dimstat$="DIM Key$[1]"
510                Dimhead=Dimhead+2*INT((LEN(Dimstat$)+1)/2)+4
520                PRINT #3;Dimstat$,END
530                GOTO Readback
540 Key2:          READ #4;Z
550                Keytype=2
560                Dimstat$="DIM Key$["&VAL$(Z)&"]"
570                PRINT #3;Dimstat$,END
580                Dimhead=Dimhead+2*INT((LEN(Dimstat$)+1)/2)+4
590                Keystat$="Key$=S"&VAL$(Varindex+1)&"$  !  "&Keytit
$
600                PRINT #6;Keystat$
610                Keyhead=2*INT(LEN(Keystat$)+1)/2+4
620                Keylength=Z
630                GOTO Readback
640 Key3:          READ #4;Z
650                Keytype=1
660                Keystat$="Key=Na"&VAL$(Varindex+1)&"("
670                Dimstat$="DIM Key$[1]"
```

```
3680                  Dimhead=Dimhead+2*INT((LEN(Dimstat$)+1)/2)+4
3690                  PRINT #3;Dimstat$,END
3700                  Keylength=8
3710                  GOTO Loopkey
3720 Key4:            READ #4;Z
3730                  Keytype=2
3740                  Keylength=Z
3750                  Dimstat$="DIM Key$["&VAL$(Z)&"]"
3760                  PRINT #3;Dimstat$,END
3770                  Dimhead=Dimhead+2*INT((LEN(Dimstat$)+1)/2)+4
3780                  Keystat$="Key$=Sa"&VAL$(Varindex+1)&"$("
3790                  READ #4;Z
3800 Loopkey:         FOR J=1 TO Z
3810                       READ #4;S,T
3820                       DISP "Please enter subscript #";J;" for the k
ey";
3830                       INPUT "",Y
3840                       IF (S<=Y) AND (Y<=T) THEN Okay2
3850                       BEEP
3860                       DISP "Subscript range is from";S;"to";T;"----
-";
3870                       GOTO 3820
3880 Okay2:                Keystat$=Keystat$&VAL$(Y)&","
3890                  NEXT J
3900                  Keystat$[LEN(Keystat$)]=")    !  "&Keytitle$
3910                  PRINT #6;Keystat$
3920                  Keyhead=2*INT(LEN(Keystat$)+1)/2+4
3930 Readback:   ! Reset the serial pointer to its position prior to en
tering                    Keyassignment
3940                  READ #4,1
3950                  FOR J=1 TO Count-1
3960                       READ #4;Y
3970                       ON Y GOSUB Read1,Read2,Read3,Read4
3980                  NEXT J
3990                  READ #4;X
4000                  RETURN


********************


4020 Read1:           RETURN
4030 Read2:           READ #4;Z
4040                  RETURN
4050 Read3:           READ #4;Z
4060                  GOTO 4080
4070 Read4:           READ #4;Y,Z
```

```
4080                FOR K=1 TO Z
4090                     READ #4;S,T
4100                NEXT K
4110                RETURN


********************


4130 Endfor: ! Titlehead=Amount of overhead for titles
4140       ! Forcount=the number format numbers in file FOR@@@
4150       ! Keyhead=Amount of overhead for key assignment
4160       ! Dimhead=Amount of overhead for DIM statements
4170       ! Prthead=Amount of overhead for PRINT# statements
4180       ! Readhead=Amount of overhead for READ# statements
4190       ! Bytesperrecord=Logical record size  (Minimum of 20 bytes)
4200       ! Poskey=Position of key's title
4210       PRINT #3;END  ! The DIM file is the only unmarked one at thi
s point
4220       ! Next, figure out how many logical records are needed for o
verhead
4230       ! by each of the categories above.
4240       Bytesperrecord=MAX(Bytesperrecord,20)
4250       Formats_per_rec=INT(Bytesperrecord/10)
4260       Titlerec=(Titlehead+4*Titlehead/Bytesperrecord)/Bytesperreco
rd
4270       IF FRACT(Titlerec)<>0 THEN Titlerec=INT(Titlerec+1)
4280       Keyrec=(Keyhead+4*Keyhead/Bytesperrecord)/Bytesperrecord
4290       IF FRACT(Keyrec)<>0 THEN Keyrec=INT(Keyrec+1)
4300       Dimrec=(Dimhead+4*Dimhead/Bytesperrecord)/Bytesperrecord
4310       IF FRACT(Dimrec)<>0 THEN Dimrec=INT(Dimrec+1)
4320       Prtrec=(Prthead+4*Prthead/Bytesperrecord)/Bytesperrecord
4330       IF FRACT(Prtrec)<>0 THEN Prtrec=INT(Prtrec+1)
4340       Readrec=(Readhead+4*Readhead/Bytesperrecord)/Bytesperrecord
4350       IF FRACT(Readrec)<>0 THEN Readrec=INT(Readrec+1)
4360       Forcount=Forcount+11 !  There are eleven extra pointers.
4370       Formats_per_rec=INT(Bytesperrecord/10)
4380       Forrec=INT(Forcount/Formats_per_rec)+NOT (NOT FRACT(Forcount
/Formats_per_rec))
4390                          ! The record format will start in record 1
4400       A=1+Forrec         ! A - position of titles
4410       B=A+Titlerec       ! B - position of PRINT# 's
4420       C=B+Prtrec         ! C - position of READ# 's
4430       D=C+Readrec        ! D - position of DIM 's
4440       E=D+Dimrec         ! E - position of key assignment
4450       F=E+Keyrec         ! F - where the name of the directory will be
 kept
```

```
4460       G=F+1              ! G - head of the free list - also total over
head
4470       PRINT PAGE;"Now you are asked to enter the name by which you
 wish"
4480       PRINT "to refer to this particular data base from now on.  P
lease"
4490       PRINT "enter the name of a file that isn't already created.
 (Note:"
4500       PRINT "file names may be up to six characters long.)"
4510       INPUT "What do you want to name your file?",Filename$
4520       ASSIGN #9 TO Filename$,Check
4530       IF Check=1 THEN 4580
4540       BEEP
4550       DISP "FILE ALREADY EXISTS"
4560       WAIT 1000
4570       GOTO 4510
4580       INPUT "What is the upper bound on the number of records you
need",Maxsize
4590       IF Maxsize<32768-G THEN 4640
4600       BEEP
4610       DISP "MAXIMUM # RECORDS IS "&VAL$(32767-G)
4620       WAIT 1000
4630       GOTO 4580
4640       PRINT PAGE;"Please wait while the file is created.  This cou
ld be"
4650       PRINT "a lengthy operation for long files on slower devices
(i.e.,"
4660       PRINT "the tape cartridge)."
4670       ON ERROR GOTO Toomuch
4680       CREATE Filename$,Maxsize+G,Bytesperrecord
4690       ASSIGN #9 TO Filename$
4700       BUFFER #9
4710       OFF ERROR
4720       GOTO 4890
4730 Toomuch:  IF ERRN=64 THEN 4760
4740 PRINT ERRM$
4750 END
4760 PRINT PAGE;"You have asked for more records than this storage med
ium has"
4770 PRINT "room for.  You may either 1) settle for fewer records, or
2) re-run"
4780 PRINT "this program using a different storage medium.  Please ent
er your"
4790 PRINT "selection of choices 1) or 2) and press CONT."
4800 INPUT "Please enter your choice of 1) or 2) and press CONT",Recov
er
4810 Recover=INT(Recover)
```

```
4820 IF (Recover<1) OR (Recover>2) THEN 4800
4830 ON Recover GOTO 4580,4840
4840 FOR I=1 TO 6
4850 PURGE Scratch$(I)
4860 NEXT I
4870 PRINT PAGE;"Please press RUN after you've installed the proper st
orage medium"
4880 END
4890       PRINT PAGE;"Please wait for all the overhead information to
be copied"
4900       PRINT "from the scratch files to the real one."
4910       IF Keytype=2 THEN Keytype=0
4920       PRINT #9;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
4930       IF Keytype=0 THEN Keytype=2
4940       ! Store pointers to the overhead areas, as well as Keytype,K
eylength,              Maxsize, and Poskey


********************


4960       ! Copy formats
4970       READ #4,1
4980       FOR I=1 TO Forcount-11
4990       READ #4;X
5000       PRINT #9;X
5010       NEXT I
5020       ! Copy titles
5030       READ #5,1
5040       READ #9,A
5050       READ #5;Title$
5060       PRINT #9;Title$
5070       IF TYP(5)<>3 THEN 5050
5080       PRINT #9;END
5090       ! Copy PRINT# 's
5100       READ #1,1
5110       READ #9,B
5120       READ #1;Prtstat$
5130       PRINT #9;Prtstat$
5140       IF TYP(1)<>3 THEN 5120
5150       PRINT #9;END
5160       ! Copy READ# 's
5170       READ #2,1
5180       READ #9,C
5190       READ #2;Readstat$
5200       PRINT #9;Readstat$
5210       IF TYP(2)<>3 THEN 5190
```

```
5220       PRINT #9;END
5230       ! Copy DIM 's
5240       READ #3,1
5250       READ #9,D
5260       READ #3;Dimstat$
5270       PRINT #9;Dimstat$
5280       IF TYP(3)<>3 THEN 5260
5290       PRINT #9;END
5300       ! Copy key assignment
5310       READ #6,1;Keystat$
5320       READ #9,E
5330       PRINT #9;Keystat$


********************


5350 ! Inititalize the free list
5360 PRINT PAGE;"Please wait for the free space on the file you have j
ust"
5370 PRINT "created to be linked together.  This may take quite a litt
le"
5380 PRINT "time for a large file on a slow device (i.e., the tape car
tridge)."
5390 IF Keytype=1 THEN Numkeys
5400 Stringkeys: Klength=Keylength
5410              Keylength=2*INT((Keylength+1)/2)+4
5420              Keyl=Keylength+(INT(Keylength/256)+1)*4
5430              GOTO 5460
5440 Numkeys: Keylength=8
5450          Keyl=10
5460 Keysinmemory=INT(2000/(Keylength+8))!Compute how many keys will f
it into 2K
5470 PRINT #9,G;G+1,Keysinmemory
5480 FOR I=G+1 TO Maxsize+G-1
5490 PRINT #9,I;I+1
5500 NEXT I
5510 PRINT #9,Maxsize+G;0


********************


5530 ! Purge all work files
5540 FOR I=1 TO 6
5550    PURGE Scratch$(I)
5560 NEXT I
```

```
********************


5580 ! Build the directory
5590 PRINT PAGE;"Now you are asked to enter a filename for the directo
ry."
5600 PRINT "The name of the directory will be stored in your other fil
e, so"
5610 PRINT "you only need remember the name of that file (";Filename$;
")."
5620 PRINT "The only reason that you are asked to enter the filename i
s so"
5630 PRINT "that the program does not inadvertantly make up the name o
f an"
5640 PRINT "already existing file."
5650 INPUT "Please enter the name of the directory",Filename$
5660 ASSIGN #1 TO Filename$,Check
5670 IF Check=1 THEN Okay3     !  Check to make sure the file does not
 exist
5680 BEEP
5690 DISP "FILE ALREADY EXISTS.  TRY AGAIN."
5700 WAIT 1000
5710 GOTO 5650
5720 Okay3: PRINT PAGE
5730 DISP "Thank you---Please wait for file initialization to be compl
eted"
5740 Dirreclength=Keysinmemory*(Keyl+8)+3*10
5750                          ! Allow room in each node for keys per rec
ord, keys
5760                          !  used in the record, an overflow number,
 the keys
5770                          !  themselves, and the record pointers.
5780 Dirnorecs=(INT(Maxsize/Keysinmemory)+1)*2
5790                          ! There should be enough nodes so that the
y're each
5800                          !  only half full when the data base is fu
ll.
5810                          !  This is to cut down insertion time.
5820 CREATE Filename$,Dirnorecs,Dirreclength
5830 ASSIGN #1 TO Filename$
5840 PRINT #9,F;Filename$
5850 CALL Initdirectory(#1,Keysinmemory,Keytype,Klength,Dirnorecs)
5860 DISP "All done initializing"
5870 BEEP
5880 END
```

```
********************


5900 SUB Initdirectory(#1,Keysinmemory,Keytype,Keylength,Dirnorecs)
5910 ON Keytype GOTO N,S
5920 N: CALL N(#1,Keysinmemory,Dirnorecs)
5930 SUBEXIT
5940 S: CALL S(#1,Keysinmemory,Keylength,Dirnorecs)
5950 SUBEXIT
5960 SUB N(#1,X,D)
5970 OPTION BASE 1
5980 DIM K(X),P(X)
5990 MAT K=(9E99)     ! Set unused keys to 9E+99
6000 MAT P=ZER        ! Set unused pointers to 0
6010 FOR I=1 TO D
6020 PRINT #1,I;X,0,0,K(*),P(*)     ! Print Dummy Records
6030 NEXT I
6040 FOR I=1 TO D
6050 READ #1,I;X,K,U,K(*),P(*)      ! Verify
6060 NEXT I
6070 SUBEXIT
6080 SUB S(#1,K,L,D)
6090 OPTION BASE 1
6100 DIM K$(K)[L],P(K)
6110 FOR I=1 TO K
6120 K$(I)=CHR$(127)     ! Set unused keys to CHR$(127)
6130 NEXT I
6140 MAT P=ZER           ! Set unused pointers to 0
6150 FOR I=1 TO D
6160 PRINT #1,I;K,0,0,K$(*),P(*)      ! Print out dummy records
6170 NEXT I
6180 FOR I=1 TO D
6190 READ #1,I;K,U,0,K$(*),P(*)       !  Verify
6200 NEXT I
6210 SUBEXIT
```

HEWLETT hp PACKARD

# DATA FILE ACCESS PROGRAMS

These programs allow the user to access his data file. Since every person's data file will probably be different, these programs write other programs which will be tailored to work with a specific data file. The information necessary to write the other programs is already stored within the user's data file by the initialization program. Thus, a user may have several data files which are not similar to each other in format, and these programs can write tailored programs to access each file.

Program Utilization:

File Names:

| | | |
|---|---|---|
| DBACCS | DBEXSN | |
| DBDRVR | DBTMP6 | These are temporary files built by the program |
| DBREPO | DBTMP1 | |
| DBLTKY | DBTMP2 | |
| DBSUB1 | DBTMP4 | |
| DBEXSS | DBTMP5 | |

Subprograms Required:

| | | |
|---|---|---|
| Report 1 | Getkey | |
| Report 2 | Findkey | |
| Sortwrite | Vectorexsort_q | |
| Listkeys | Stringexsort_q | |
| Findemptyrecord | Replacekey | These subprograms are written by other programs |
| Returnrecord | Merge | |
| Update | Enter | |
| Binsearch | Edit | |
| Garbagecollect | Report | |
| Hashpage | | |

Variables (Program construction program):

A - The record number in the data file where the titles of the data fields start

A$ - Used to enter file names and mass storage devices. Also used for intermediate storage of program lines as they are being written.

B - The record number in the data file where the PRINT # statement(s) start

B$ - Used to write program lines out to mass storage

C - The record number in the data file where the READ # statement(s) start

C$ - Used to hold the key assignment statement

Check - Used to test for file types in ASSIGN statements

D - The record number in the data file where the DIM statement(s) start

D$ - Used to help build DISP statements for prompts

Device$ - Holds the device type of the mass storage unit holding the user's programs

E - The record number in the data file where the key assignment statement is kept

F - The record number in the data file where the name of the directory file is kept

File$ - The data file

G - The record number in the data file where the head of the free list is. Also, the total number of overhead records required

H$ - Holds a subscript string for help in building variable names and labels

| | | |
|---|---|---|
| I | - | Keeps track of line number |
| I$ | - | Used to help build INPUT statements |
| J | - | Loop counter |
| J$ | - | Holds a subscript string for help in building prompts for entering arrays |
| Keydim$ | - | Holds the DIM statement where the key is dimensioned, if the key is a string |
| Keyflag | - | 0 if the key is a string, 1 if the key is numeric |
| Keylength | - | The length of the key |
| Keytype | - | 0 if the key is a string, 1 if the key is numeric |
| L | - | Temporary storage for string length |
| Label$ | - | Helps in building IF and GOTO statements for branching to a certain label (Label$ is either "m" or "s") |
| Maxsize | - | The maximum number of data records available to the user in his data file |
| Nonummats | - | The number of numeric arrays in each data record |
| Nonums | - | The number of simple numeric fields in each data record |
| Nostrmats | - | The number of string arrays in each data record |
| Nostrs | - | The number of simple string fields in each data record |
| Overlay | - | Tells whether or not it is necessary to overlay programs due to memory size (Overlay=1 means that the 32K option is being used, Overlay=0 means that the 64K option is being used) |
| Pbl | - | The position of the first blank space in the key assignment statement |
| Pes | - | The position of the "=" in the key assignment statement |

Plp - The position of the left parentheses "(" in the key assignment statement (if there is one)

Poskey - The position of the key field of the data record with respect to the other fields in the record

Prp - The position of the right parentheses ")" in the key assignment statement (if there is one)

S$ - Used to pad out the beginning of the program lines with blanks

S(*) - Temporary storage for array dimension bounds

Subprompts$ - Used for building a set of subscripts for prompts

Subscripts$ - Used for building a set of subscripts for I/O statements

Title$ - Holds the titles of the fields of the data record

Type$ - Either "string" or "numeric". Used for prompts

Varnam$ - Variable name. Used for INPUT's and EDIT's

X - Temporary storage for format codes

Variables (Driver):

A$ - Used for the user's answers to Y/N questions

Already_saved - Tests to see if a report is already saved, so the user is not asked if he wants to save an already saved report format

C - Used with ASSIGN statements to test for file types

Cnumber - Command number. Used with 32K configuration to test to see if a subroutine needs to be linked in, or if it is already in memory

Collide - Tells whether or not the user has entered a key that has already been used

Command$ - This string holds the commands that the user issues

Device$ - The mass storage unit where the programs are stored

Dirname$ - The name of the directory file

Dirnorecs - The number of records in the directory file

Key - Holds the key if it is numeric

Key$ - Holds the key if it is a string

Keyflag - Tells whether the key is a number or a string (0 if string, 1 if numeric)

Keylength - The length of the key if it is a string

Keysinmemory - The maximum number of keys (with their associated pointers that will fit into a 2K byte slice of memory)

Maxsize - The maximum number of data records available to the user

Overlay - Tells whether or not the programs need to be overlayed due to memroy size (0 implies a 64K configuration, 1 implies a 32K configuration)

Pageno - The record number of the directory file where the current key belongs

Recno - The record number in the data file where the current key's record may be found

Reportfile$ - The name of the file which holds the report and merge subprograms that the user specifies

Sortlength - The length of the field that the user has specified print reports on

X - Dummy variable

Variables (Subprogram Report1):

A - The record number in the data file where the titles of the fields in the data record start

Answer$ - Holds the user's answers to Y/N questions

B - The record number in the data file where the PRINT # statement(s) start

| | | |
|---|---|---|
| C | - | The record number in the data file where the READ # statement(s) start |
| Colperline | - | The number of columns per line on the printer to be used for the report |
| Count | - | Counts the number of fields in the data record |
| Count$ | - | Used to print the numbers of the fields in the data records |
| D | - | The record number in the data file where the DIM statement(s) start |
| Device | - | The HPIB device address of the printer used for the report |
| E | - | The record number in the data file where the key assignment statement is stored |
| F | - | The record number in the data file where the name of the directory file is kept |
| G | - | The record number in the data file where the head of the free list is (also the total number of overhead records used) |
| Howmanyfields | - | The number of fields that the user wants in his report |
| I | - | Loop counter |
| Keylength | - | The length of the key field (in bytes) |
| Keytype | - | The type of the key (0 if string, 1 if numeric) |
| L | - | Temporary storage for string length |
| Maxsize | - | The maximum number of data records available to the user |
| Poskey | - | The position of the key fields with respect to the rest of the fields in the data record |
| Select | - | The select code of the printer which is used to print the report |

Sortlength - The length of the field (in bytes) on which the report will be sorted

Sorttype - The type of the field on which the report will be sorted

Title$ - Holds the titles of the fields in the data record

Titlemode - Allows for two different types of titles on the report
1 - The titles will occur once at the top of the report
2 - The titles will occur before each data record

X - Temporary storage for format codes

Y - Temporary storage for format codes

Z - Temporary storage for format codes

Variables (Subprogram Report2):

A - The record number in the data file where the titles of the fields in the data record start

A$ - Used for the user's answers to Y/N questions. Also used as temporary storage for building program statements

After - Used in building IMAGE statements. Tells how many blanks should be padded in after a field is built

B - The record number in the data file where the PRINT # statement(s) start

B$ - Used in building program statements

C - The record number in the data file where the READ # statement(s) start

Check - Used in conjunction with ASSIGN statements to see if a data file already exists

Colperline - The number of columns per line on the printer being used for the report

| | | |
|---|---|---|
| Count | - | The total number of fields in the data record |
| D | - | The record number in the data file where the DIM statement(s) start |
| Device | - | The HPIB device address (if any) of the printer being used for the report |
| E | - | The record number in the data file where the key assignment statement is stored |
| Endsub | - | Used in printing out strings that are longer than the printer width. Endsub gives the column number of the end of the substring which will fit on the line. |
| F | - | The record number in the data file where the name of the directory file is stored |
| G | - | The record number in the data file where the head of the free list is kept |
| Howmanyfields | - | The number of fields that the user wants in his report |
| I | - | Loop counter |
| Image$ | - | Used to build IMAGE statements for the report |
| Imagecounter | - | Count the number of IMAGE statements needed for each record |
| Imaget$ | - | Used to build IMAGE statements for the titles of the report |
| J | - | Loop counter |
| K | - | Loop counter |
| Keylength | - | The length (in bytes) of the key field |
| Keytype | - | The type of the key field (0 if string, 1 if numeric) |
| L | - | Temporary storage for string length |
| Linenumber | - | The current line number of the program being written |

| | | |
|---|---|---|
| Maxsize | - | The maximum number of data records available to the user |
| Nonummats | - | The number of numeric arrays in the user's data record |
| Nonums | - | The number of numeric fields in the user's data record |
| Nostrmats | - | The number of string arrays in the user's data record |
| Nostrs | - | The number of string fields in the user's data record |
| P | - | Used for padding IMAGE statements with blanks |
| Placesafter | - | The number of decimal places after a decimal point in a numeric field |
| Placesbefore | - | The number of decimal places before the decimal point in a numeric field |
| Poskey | - | The position of the key field with respect to the other fields in the data record |
| Print$ | - | Used to build the PRINT statements for the report |
| Printt$ | - | Used to build the PRINT statements for the titles in the report |
| Roomleft | - | Tells how much room is left in the line |
| Roomneeded | - | Tells how much room is needed for the next field |
| S(*) | - | Temporary storage for array subscripts |
| Select | - | The select code of the printer to be used in the report |
| Sortlength | - | The length (in bytes) of the field which is used to sort on |

Sorttype - The type of the field to be sorted on (1 for numeric, 2 for string)

Startsub - The column where the substring which will fit on one line starts

Title$ - The titles of the fields in the data records

Titlecounter - Counts the number of lines in the titles

Titlemode - 1 for one set of titles, 2 for a set of titles above every record

Tlinestart - The current line number of the subroutine in which the titles are printed

Usedflag - Tells whether or not the last PRINT and IMAGE statements have been used

Varnam$ - Variable name (used in building PRINT statements for the report)

Whichfields(*) - This array tells which fields are to be used in the report

X - Temporary variable for format codes

Y - Temporary variable for format codes

Variables (Subprogram Sortwrite):

A - The record number in the data file where the titles of the fields of the data record start

B - The record number in the data file where the PRINT # statement(s) start

B$ - Used to build program statements for the merge routine

C - The record number in the data file where the READ # statement(s) start

Check - Used with ASSIGN statements to see whether or not a dummy file is available

Count - Counts the fields in the user's data record

D - The record number in the data file where the DIM statement(s) start

Dim$ - Used to hold DIM statements as they are read from the data file and used to write the merge routine

Dimstat$ - Used to build the DIM statement used for the sort field

E - The record number in the data file where the key assignment statement starts

F - The record number in the data file where the name of the directory file starts

G - The record number in the data file where the head of the free list is (also the total number of records needed for overhead)

I - Loop counter

J - Loop counter

K - Loop counter

K1 - Dummy variable

K2 - Dummy variable

K3 - Dummy variable

K4 - Dummy variable

Keyflag - The type of the key field (0 for string, 1 for numeric)

Pos_sort - The position of the sort field with respect to the other fields in the data record

Readstat$ - The string used to hold the READ # statement(s) as they are written into the merge routine

Recordlength - The record length of the merge file

S - Dummy variable

Sortlength - The length (in bytes) of the field being sorted on

Sortstat$ - The sort field assignment statement

Sorttitle$ - The title of the sort field

Sorttype - The type of the sort field (1 if numeric, 2 if string)

Statno - The statement number

T - Dummy variable

Title$ - Holds the titles of the fields of the data records

Varindex - The index of the variable

X - Temporary storage for format codes

Y - Temporary storage for format codes

Z - Temporary storage for format codes

Variables (Subprogram Findemptyrecord):

Freepoint - The record number in the data file where the head of the free list is kept

Nextpoint - The link field of the first record in the free list

Recno - The record number of the first record in the free list

X - Dummy variable (the maximum number of keys in memory)

Variables (Subprogram Returnrecord):

Freepoint - The record number in the data file where the head of the free list is kept

Nextpoint - The old first record in the free list (now the second free record in the free list)

Recno - The record number of the record being returned to the free list

X - Dummy variable (the maximum number of keys in memory)

Variables (Subprogram Updatepage)

| | | |
|---|---|---|
| Collide | - | Tells whether or not a key has already been used (1 is collision, 0 is no collision) |
| Dirnorecs | - | The number of records in the directory file |
| Key | - | Holds the key value if it is a number |
| Key$ | - | Holds the key value if it is a string |
| Keyflag | - | Tells whether the key is a numeric or a string (0 for string, 1 for numeric) |
| Keylength | - | The length (in bytes) of the key field |
| Keysperrecord | - | The number of keys which will fit into a single directory file record |
| Pointer | - | The record number of the data record having the given key value |
| Recno | - | The record number of the directory file record which contains the given key value |

Variables (Subprogram Upstring):

| | | |
|---|---|---|
| Collide | - | Tells whether or not the given key value has already been used |
| Dirnorecs | - | The number of records in the directory file |
| Key | - | Holds the key value if it is a number |
| Key$ | - | Holds the key value if it is a string |
| Key$(*) | - | Array holding all the keys in a given directory record if the key field is a string |
| Key(*) | - | Array holding all the keys in a given directory record if the key field is numeric |
| Keyflag | - | Tells whether the key value is a number or a string |
| Keylength | - | Gives the length (in bytes) of the key field |
| Keysperrecord | - | The number of keys which will fit into a single directory file record |

Pointer - The record number of the data record having the given key value

Pointers(*) - The array which holds all the record pointers in a given directory file record

Recno - The record number of the directory file record which contains the given key value

Variables (Subprogram Upnum):

Collide - Tells whether or not the given key value has already been used

Dirnorecs - The number of records in the directory file

Key - Holds the key value if it is a number

Key$ - Holds the key value if it is a string

Key$(*) - Array holding all the key values in a directory record if the key field is a string

Keyflag - Tells whether the key value is a numeric or a string

Key(*) - Array holding all the key values in a directory record if the key field is numeric

Keylength - The length (in bytes) of the key field

Keysperrecord - The maximum number of keys that will fit into a directory file record

Pointer - The record number of the data record having the given key value

Pointers(*) - The array which holds all the record pointers in a given directory file record

Recno - The record number of the directory file record which contains the given key value

Variables (Subprogram Update):

| | | |
|---|---|---|
| Collide | - | Tells whether or not the given key value has already been used |
| Dirnorecs | - | The number of records in the directory file |
| Found | - | Tells whether or not the given key value has been found in the specified directory record Recno (0 is not found, 1 is found) |
| I | - | Loop counter |
| Key | - | Holds the key value if it is a number |
| Key$ | - | Holds the key value if it is a string |
| Keyflag | - | Tells whether the key field is a number or a string (0 for a string, 1 for a number) |
| Keylength | - | Tells the length (in bytes) of the key field |
| Keys$(*) | - | Holds all the key values in a directory record if the key field is a string |
| Keys(*) | - | Holds all the key values in a directory record if the key field is a number |
| Keysperrecord | - | The maximum number of keys that will fit into a directory file record |
| Keysused | - | Tells how many keys there actually are in a directory record |
| Overflow | - | Tells whether or not the current directory record has ever overflowed (0 for no overflow, 1 for overflow) |
| Pointer | - | The location of the data record associated with the given key value |
| Pointers(*) | - | Array holding all the record pointers in a given directory record |
| Recno | - | The record number of the directory file record which should contain the given key value |

Where - If the given key value has been found in the directory file record, Where tells where in the record the key was found

Variables (Subprogram Garbagecollect):

D - The subscript of the key value to be deleted

Deletesubscript- The subscript of the key value to be deleted

I - Loop counter

Keyflag - Tells whether the key field is a number or a string

Keys$(*) - Array holding the keys in a directory file record if the key field is a string

Keys(*) - Array holding the keys in a directory file record if the key field is numeric

Keysperrecord - The maximum number of keys that will fit into a directory file record

Keysused - The number of keys actually in a directory file record

Pointers(*) - Array holding the record numbers of the data records associated with the key values

Variables (Subprogram Hashpage):

H - Temporary variable

Hold - Temporary variable

I - Loop counter

Key - The key value being hashed (if the key field is numeric)

Key$ - The key value being hashed(if the key field is a string)

Keyflag - Tells whether the key to be hashed is a number or a string

Keylength - Tells the length (in bytes) of the key field

Norecs - The number of records in the directory file

Pageno - The directory file record which will contain the given key value

Step - Loop step

Variables (Subprogram Getkey):

A - The record number in the data file where the titles of the fields in the data record start

B - The record number in the data file where the PRINT # statement(s) start

C - The record number in the data file where the READ # statement(s) start

D - The record number in the data file where the DIM statement(s) start

Delete_or_edit$ - A string having the value "delete" or "edit" which is used in a prompt to the user (this variable is necessary because this routine is used to enter a key for both deletion and edition)

E - The record number in the data file where the key assignment statement is stored

F - The record number in the data file where the name of the directory file is kept

G - The record number in the data file where the head of the free list is kept (also the total number of overhead records in the data file)

I - Loop counter

Key - Holds the key value if it is numeric

Key$ - Holds the key value if it is a string

Keyflag - Tells whether the key field is a number or a string

Keylength - Tells the length (in bytes) of the key field

Keytype - Dummy variable

Maxsize - The maximum number of data records available to the user

Poskey - The position of the key field with respect to the other fields in the data record

T$ - Holds the titles of the fields in the data record

Type$ - This string either has the value "string" or "numeric" and is used to inform the user (via a prompt) what kind of value he is expected to enter

Variables (Subprogram Findkey):

Dirnorecs - The number of records in the directory file

Key - Holds the key value if it is a number

Key$ - Holds the key value if it is a string

Keyflag - Tells whether the key field is numeric or string

Keylength - Gives the length (in bytes) of the key field

Keysinmemory - The maximum number of keys that will fit in a directory file record

Maxsize - The maximum number of data records available to the user

Pageno - The directory file record number where the key value belongs

Switch - Tells whether the key should be inserted in the directory record or deleted from the directory record (0 is delete, 1 is insert)

Variables (Subprogram Fs):

| | | |
|---|---|---|
| Dirnorecs | - | The number of records in the directory file |
| Key | - | Holds the key value if it is a number |
| Key$ | - | Holds the key value if it is a string |
| Keyflag | - | Tells whether the key field is numeric or string |
| Keylength | - | Gives the length (in bytes) of the key field |
| Keys$(*) | - | Holds all the key values in a directory file record if the key field is a string |
| Keys(*) | - | Holds all the key values in a directory file record if the key field is numeric |
| Keysinmemory | - | The maximum number of keys that will fit in a directory file record |
| Maxsize | - | The maximum number of data records available to the user |
| Pageno | - | The directory file record number where the key value belongs |
| Pointers(*) | - | Array holding the record pointers associated with the key values in either array Keys(*) or Keys$(*) |
| Switch | - | Tells whether the key should be inserted in the directory record or deleted from the directory record (0 is delete, 1 is insert) |

Variables (Subprogram Fn):

| | | |
|---|---|---|
| Dirnorecs | - | The number of records in the directory file |
| Key | - | Holds the key value if it is a number |
| Key$ | - | Holds the key value if it is a string |
| Keyflag | - | Tells whether the key field is numeric or string |
| Keylength | - | Gives the length (in bytes) of the key field |
| Keys$(*) | - | Holds all the key values in a directory file record if the key field is a string |
| Keys(*) | - | Holds all the key values in a directory file record if the key field is numeric |
| Keysinmemory | - | The maximum number of keys that will fit in a directory file record |

Maxsize - The maximum number of data records available to the user

Pageno - The directory file record number where the key value belongs

Pointers(*) - Array holding the record pointers associated with the key values in either array Keys(*) or Keys$(*)

Switch - Tells whether the key should be inserted in the directory record or deleted from the directory record (0 is delete, 1 is insert)

Variables (Subprogram Binsearch_s):

Dim - The dimension of the increment array (also the maximum number of searches necessary)

Found - Tells whether or not the key is found (0 for not found, 1 for found)

Key$ - Holds the value of the key

Keys$(*) - The array being searched for containing Key$

Keysused - The number of keys in the array Keys$(*)

Where - The position of the found key, or the position immediately below the position where the key would belong if it's not there

Variables (Subprogram Bs):

Delta(*) - Array containing the increments in the binary search

Dim - The dimension of Delta(*) (also the maximum number of tests necessary to tell whether or not the key exists)

Found - Tells whether or not the key was found

I - Current index into the array

J - Current index into the increment array

Key$ - The key value being sought

Keys$(*) - The array being searched

Keyused - The number of elements in Keys$(*)

Where - The position of the found key, or if the key is not found, the position immediately below where the key would go if it was found

Variables (Subprogram Binsearch_n):

Dim - The dimension of the increment array (also the maximum number of searches necessary)

Found - Tells whether or not the key was found (Ø for not found, 1 for found)

Key - Holds the value of the key

Keys(*) - The array being searched for containing Key$

Keysused - The number of keys in the array Keys$(*)

Where - The position of the found key, or the position immediately below the position where the key would belong if it is not there

Variables (Subprogram Bn):

Delta(*) - Array containing the increments in the binary search

Dim - The dimension of Delta(*)(also the maximum number of tests necessary to tell whether or not the key exists)

Found - Tells whether or not the key was found

I - Current index into the array

J - Current index into the increment array

Key - The key value being sought

Keys(*) - The array being searched

Keysused - The number of elements in Keys$(*)

Where - The position of the found key, or if the key is not found, the position immediately below where the key would go if it was found

Variables (Subprogram Stringexsort_q):

| | | |
|---|---|---|
| Il | - | Lower bound of the external sort |
| Jl | - | Upper bound of the external sort |
| Length | - | The length of the strings being sorted |
| Logtwo | - | The log (base two) of the number of records being sorted |
| N | - | The number of records being sorted |

Variables (Subprogram Qsortl)

| | | |
|---|---|---|
| Ai$ | - | Holds the value of the ith record |
| Aj$ | - | Holds the value of the jth record |
| Ak$ | - | Holds the value of the kth record |
| Al$ | - | Holds the value of the lth record |
| I | - | Lower endpoint of file segment being sorted |
| Il | - | Lower bound of sort |
| I2 | - | Middle element of file segment being sorted |
| J | - | Upper endpoint of file segment being sorted |
| Jl | - | Upper bound of sort |
| K | - | Temporary storage for I |
| L | - | Temporary storage for J |
| L(*) | - | Stack for lower endpoints of file segments |
| Length | - | The length of strings being sorted |
| Log | - | The dimension of the arrays L(*) and U(*) (Log2 or the number of records to be sorted) |
| M | - | Stack pointer |

Pi - Pointer field of the ith record

Pj - Pointer field of the jth record

Pk - Pointer field of the kth record

Pl - Pointer field of the lth record

Pt - Temporary storage for switching pointers

Ptl - Temporary storage for switching pointers

T$ - Temporary storage for switching elements

Tl$ - Temporary storage for switching elements

U(*) - Stack for upper endpoints of file segments

Variables (Subprogram Vectorexsort_q):

Il - Lower bound of the external sort

Jl - Upper bound of the external sort

Logtwo - The log (base two) of the number of records being sorted

N - The number of records being sorted

Variables (Subprogram Qsortl):

Ai - Holds the value of the ith record

Aj - Holds the value of the jth record

Ak - Holds the value of the kth record

Al - Holds the value of the lth record

I - Lower endpoint of file segment being sorted

Il - Lower bound of sort

I2 - Middle element of file segment being sorted

J - Upper endpoint of file segment being sorted

Jl - Upper bound of sort

K - Temporary storage for I

L - Temporary storage for J

L(*) - Stack for lower endpoints of file segments

Log - The dimension of the arrays L(*) and U(*) (Log2 of the number of records to be sorted)

M - Stack pointer

Pi - Pointer field of the ith record

Pj - Pointer field of the jth record

Pk - Pointer field of the kth record

Pl - Pointer field of the lth record

Pt - Temporary storage for switching pointers

Ptl - Temporary storage for switching pointers

T - Temporary storage for switching elements

Tl - Temporary storage for switching elements

U(*) - Stack for upper endpoints of file segments

Special Considerations and Programming Hints:

1. As there is extensive linking between programs, and large amounts of mass storage access going on during this part of the system's operation, it becomes desirable to use a disk device for mass storage, rather than a tape cartridge. There are two reasons for this: 1) Speed. Disk devices do not require the rewind or search time that the tape cartridge does. 2) Wear. The constant tape movement required by this system could wear out a tape in a relatively short period of time.

2. To run with the standard memory not all of the subprograms may reside in memory at once. Formatting a new report form requires a different set of subprograms than the other operations. The appropriate subprograms will be linked in as they are needed.

3. The user may choose the order in which his reports will print the data fields to a certain extent. The program will find all the numeric fields and string fields which will fit on one line in the order that the user specifies them first. Then the program will print any string fields that do not fit on one line (in the order the user specifies), and last, any arrays will be printed (in the order the user specifies).

User Instructions:

1. Insert the storage medium containing the Information Management Routines into the mass storage device you wish to use for the programs.

2. Load the file:
   a. Type: LOAD "DBACCS:(Mass Storage Unit Specifier) where (Mass Storage Unit Specifier) defines which device you are loading the programs from (examples: To load from the primary tape transport, type LOAD DBACCS:T15". To load from a floppy on select code 8, type LOAD "DBACCS:F8".)
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Please enter the device code where your data is stored and press CONT" appears in the display:
   a. Type: (Mass Storage Unit Specifier) where (Mass Storage Unit Specifier) defines which device is being used to hold your data (examples: If you are using a floppy disk on select code 8 for your data, type F8. If you are using a 7905 disk drive on select code 12, type Y12.)
   b. Press: CONT

5. When "Please enter the device code where your programs are stored" appears in the display area:
   a. Type: (Mass Storage Unit Specifier) where (Mass Storage Unit Specifier) defines which device will be used to load your programs from (this should be the same (Mass Storage Unit Specifier) you used in step 2).
   b. Press: CONT
6. When "Now, please enter the name of the data file you wish to access" appears in the display area:
   a. Type: The name of the file you wish to access
   b. Press: CONT
7. When "Does your machine have a memory larger than 64K (Y/N)?" appears in the display area of the CRT:
   a. If your machine has more memory than the standard memory opti
      1) Type: Y
      2) Press: CONT
      3) Go to step 8.

      or
   a. If you have the standard memory option:
      1) Type: N
      2) Press: CONT
      3) Go to step 8.

At this point the program will read certain information that was stored in your data file when it was initialized, and use that information to write several routines which will be tailored to work with your data file. These subprograms will allow you to 1) enter data to your file, 2) edit data from the file, 3) replace the key of a record in the event that the key is not unique, and 4) merge all the keys and pointers from the directory onto a single file where they can be sorted. In addition, a specialized driver program is also written, which takes into account what kind of key field the data file used, how large the data file is, how large the memory of the machine is and so on. This process can take several minutes, so do not be alarmed at the delay.

8. When "Command?" appears in the display area of the CRT:
   a. Type: A valid command (the valid commands will be printed on the CRT)
   b. Press: CONT
      (Note: The keywords and their meanings are given below.
      ENTER - Enter a data record from the keyboard.
      LIST KEYS - List all of the values of the key fields of each record in the data base.
      EDIT - Edit a data record from the keyboard.
      EDIT LAST - Edit the data record having the key value most recently used by some other command (Either ENTER, EDIT, EDIT LAST, or DELETE).
      DELETE - Delete a record from the data base.
      REPORT - Print out a report on the data base.
      STOP - End the program.)
9. Depending upon the command entered in part 8a, go to one of the following steps:
   a. If you entered the command ENTER goto step 10.
   b. If you entered the command LIST KEYS go to step 16.
   c. If you entered the command EDIT go to step 13.
   d. If you entered the command EDIT LAST go to step 14.
   e. If you entered the command DELETE go to step 17.
   f. If you entered the command REPORT go to step 19.
   g. If you entered the command STOP, the message "PROGRAM TERMINATED" will appear at the top of the CRT, and the program will be finished.
   h. If you type in a command that the system does not recognize, the message "XXXX is an invalid command. Please try again." will be displayed briefly, and then the prompt "Command?" will reappear. Go back to step 8a and enter a valid command.
10. If the message "FILE IS FULL. THERE IS NO MORE ROOM." is printed on the CRT, then you have used up all the room in the file. No more records can be entered until something is deleted first. The other commands can still be used, how-

ever. In this event, you must return to step 8.

11. Repeat this step (step 11) as often as necessary.
    a. As a prompt to enter a value appears in the display area of the CRT, along with the title of the field for which you are entering data:
        1) Enter: The requested value
        2) Press: CONT
    b. After the last value in the data record has been entered, go to step 12.

12. a. If the value of the key field of the data record entered during the previous step is already being used by another record:
        1) When "Please enter a new ["string" or "numeric"] value for XXXX" (where XXXX is the title of the key field) appears in the display area:
            a) Enter: Any value for the key that is not being used by another data record
            b) Press: CONT
            c) If the value of the key entered in 12a1)a) is unused elsewhere, then go to step 8. If the value <u>is</u> one being used elsewhere, you will be forced to repeat step 12a1).

    or

    a. If the value entered in the key field of the record was not being used in another data record:
        1) Go to step 8.

13. When "Please enter the ["string" or "numeric"] value for the key" appears in the display area of the CRT:
    a. Enter: The value of the key of the record you wish to edit
    b. Press: CONT
    c. Next you will be given the chance to verify that you have not mistyped the key value you entered in step 13a. Make sure you entered the right key and proceed to step 13d.
    d. If the key exists, go on to step 14.

or

d. If the key does not exist, the message "There is no record having the key XXXX" will be printed. Go to step 8.

14. Repeat this step (step 14) as often as necessary.

a. The title of each field and its old value will be printed on the CRT one by one, and a corresponding prompt will appear in the display area of the CRT. At this point, you may do one of two things:

1) If you wish to change the value of the indicated field:

a) Enter: A new value for the field

b) Press: CONT

or

2) If you wish to retain the old value of the indicated field:

a) Press: CONT

b. If the displayed field happens to be an array (either numeric or string), the message "Do you wish to change anything under the array XXXX (Y/N)?" will appear in the display area.

1) If you wish to change anything under the given array:

a) Type: Y

b) Press: CONT

c) Go to step 14b2).

or

1) If you do not wish to change anything under the given array:

a) Type: N

b) Press: CONT

c) Repeat step 14 as needed.

2) When the message "Please enter the mode you wish to use (1 or 2)" appears in the display:

a) If you want to edit every element of the array, one by one:

i) Enter: 1

ii) Press: CONT

iii) Go to step 14b3).

or

a) If you want to edit only one element of the array:

i) Enter: 2

ii) Press: CONT

iii) Go to step 14b4).

3) As the subscripts for each element of the array are displayed, you may do one of two things:

a) If you wish to enter a new element at the indicated position of the array:

i) Enter: A new array element

ii) Press: CONT

or

a) If you wish to retain the old value of the indicated array element:

i) Press: CONT

b) Repeat step 14b3) until each of the array elements has been edited.

c) Repeat step 14 as needed.

4) When "Dimension # j (Range: x TO y)?" appears in the display area of the CRT:

a) Enter: The jth subscript of the element of the array which you wish to change

b) Press: CONT

c) Repeat step 14b4) for each dimension of the array.

5) When "Please enter the value for XXXX(a,b,c,...etc)" appears in the display area of the CRT:

a) Enter: The new value for the indicated element of the array.

b) Press: CONT

c) Repeat step 14 as needed.

c. When all of the fields of the data record have been edited, go to step 15.

15. a. If you altered the value of the key field during the editing procedure, and if the new value of the key is already being

used by another data record:

1) When "Please enter a new ["string" or "numeric"] value for XXXX" (where XXXX is the title of the key field) appears in the display area:
   a) Enter: Any value for the key that is not being used by another data record
   b) Press: CONT
   c) If the value of the key entered in 15a1)a) is unused elsewhere, then go to step 8. If the value is one being used elsewhere, you will be forced to repeat step 15a1).

   or

a. If the field being used for the key was unchanged during the editing procedure, or if the key field was changed but the new value did not conflict with the key value of another record:
   1) Go to step 8.

16. The title "KEYS:" will be printed on the CRT, followed by the keys of all the records in the data base. When this process is complete, go to step 8.

17. When "Please enter the ["string" or "numeric"] value for the key" appears in the display area of the CRT:
   a. Enter: The value of the key of the record you wish to delete
   b. Press: CONT
   c. Next you will be given the chance to verify that you have not mistyped the key value you entered in step 17a. Make sure you entered the right key and proceed to step 17d.
   d. If the key exists, go on to step 18.

   or

   d. If the key does not exist, the message "There is no record having the key XXXX" will be printed. Go to step 8.

18. When the record has been deleted, the message "The record having the key XXXX has been deleted." will be printed on the CRT. After this has happened go to step 8.

19. When "Please enter the title of the field you wish to sort on" appears in the display area:
    a. Type: The title of the field you wish to sort on
    b. Press: CONT
    c. If the field whose title you entered in step 19a is a simple numeric or a simple string field, then go on to step 20. If the field was a numeric or string array, however, go to step 19d.
    d. When "Please enter subscript #j for the sort field" appears in the display area:
        1) Enter: The subscript of the jth dimension
        2) Press: CONT
        3) Repeat step 19d for each subscript of the array.
20. When "Do you want the report on an HPIB printer (Y/N)?" appears in the display area of the CRT:
    a. If you are using an HPIB printer:
        1) Type: Y
        2) Press: CONT
    b. When "Please enter the HPIB address of the printer" appears in the display:
        1) Enter: The HPIB address (which is set on the back of the printer)
        2) Press: CONT

    or

    a. If the printer you are using is not an HPIB device:
        1) Type: N
        2) Press: CONT
21. When "Please enter the select code of the printer" appears in the display area of the CRT:
    a. Enter: The select code of the printer on which you wish to print the report (Note: The select code of the internal printer on the 9845A is Ø, and the select code of the CRT is 16).
    b. Press: CONT

c. When "How many columns per line does the printer have?" appears in the display area:

1) Enter: The number of columns per line that the printer has (or the number of columns that you want the report to use). Note: The valid range for the number of columns on the printer is from 40 to 132.

2) Press: CONT

22. When "Please enter 1 or 2 and press CONT to select the title mode you want" appears in the display area:

a. If you want the titles of the field to be printed out once at the very top of the report:

1) Enter: 1

2) Press: CONT

3) Go to step 23.

or

a. If you want the titles to be printed above every data record on the report:

1) Enter: 2

2) Press: CONT

3) Go to step 23.

23. The titles of each of the fields of the data record will be printed on the CRT. When "How many of the above fields do you wish to have in your report?" appears in the display area of the CRT:

a. Enter: The number of fields you wish to include in the report

b. Press: CONT

24. If you entered a number equal to the total number of fields in the data record in step 23a, perform step 24a. If the number you entered in step 23a is less than the total number of fields available, go to step 25.

a. When "Do you want to pick the order the fields will occur in (Y/N)?" appears in the display area:

1) If you wish to pick the order in which the fields will occur on the report:

a) Type: Y
b) Press: CONT
c) Go to step 25.

or

1) If you want the fields to occur in the same order as they are listed on the screen:
   a) Type: N
   b) Press: CONT
   c) Go to step 26.

25. When "Please enter the title number of the next field?" appears in the display area:
    a. Enter: The title number of the next field that you want to include in your report (Note that a field may appear more than once in the report.)
    b. Press: CONT
    c. Repeat step 25 for every field that you want in the report. Note: The program will format the report so that all arrays and strings whose length exceeds the printer width are printed after simple numeric fields and short string fields.
26. Now the program will write a subprogram that will output the report in the way you have requested. For any numeric field in the report you must perform step 27. Otherwise, wait until the report is finished. Then go back to step 8.
27. If "Please enter the format code (1 or 2)" appears in the display area:
    a. If you wish to use fixed point format for the indicated numeric field (the title of the field will be printed on the CRT):
       1) Enter: 1
       2) Press: CONT
       3) Go to step 27b.

       or

    a. If you wish to use floating point format for the numeric field in question:

1) Enter: 2

2) Press: CONT

3) Go to step 24c.

b. When "How many places before the decimal point should be allowed for?" appears in the display area:

1) Enter: The number of digits you want to appear before the decimal point (Note that if you have numbers that have more places before the decimal point than you allow for here, they will not fit the IMAGE statement that this program is preparing and you will get garbage in the report. Also note that you need not allow for the sign of the number as the program automatically handles that. You are restricted to 12 places before the decimal point. If your numbers are greater than 1E+12 then use floating point formats.)

2) Press: CONT

c. When "How many places after the decimal point should be allowed for?" appears in the display area of the CRT:

1) Enter: The number of digits that you want to occur after the decimal point (Note that the maximum number of digits after the decimal point is 12. If you are using floating point format, that is all the precision this machine carries. If you are using fixed point format and you have numbers with more than 12 digits of precision, you should be using floating point format anyway. Again, the signs are taken care of automatically.)

2) Press: CONT

d. Go back to step 26.

Listing of Program Construction Program (file "DBACCS")

```
10   OPTION BASE 1
20   DIM A$[160],B$[160],C$[160],S(6,2),I$[80],D$[80],Subprompts$[30],Keydim$[80
],Title$[30]
30   PRINTER IS 16
40   PRINT PAGE
50   PRINT "Please enter the device code of the mass storage device you are usin
g,"
60   PRINT "both for storing your data, and for where your programs are stored."
70   PRINT "(Examples:  If you are using the primary tape cartridge, enter T15."
80   PRINT "If you are using a floppy on select code 8, enter F8.  If you are us
ing"
90   PRINT "a hard disk on select code 10, enter D10.)"
100  LINPUT "Please enter the device code where your data is stored and press
CONT",B$
110  ON ERROR GOTO Bomb
120  MASS STORAGE IS ":"&B$
130  LINPUT "Please enter the device code where your programs are stored and p
ress CONT",A$
140  Device$=":"&A$
150  MASS STORAGE IS Device$
160  OFF ERROR
170  MASS STORAGE IS ":"&B$
180  GOTO 230
190 Bomb:  BEEP
200  DISP "ILLEGAL. START OVER."
210  WAIT 1000
220  GOTO 100
230  PRINT PAGE
240  S$="     "
250  INPUT "Now, please enter the name of the data file you wish to access",A$
260  ASSIGN #1 TO A$,Check
270  IF Check=0 THEN 320
280  BEEP
290  DISP "FILE ";A$;" DOES NOT EXIST ON THE SPECIFIED DEVICE"
300  WAIT 1200
310  GOTO 40
320  PRINT PAGE;"The following information is needed to determine how much of th
e "
330  PRINT "program can fit into memory at once.  If you have a 64k machine then
"
350  PRINT "routines will have to be loaded into memory as they are needed."
```

```
360  PRINT "On the other hand, if your machine is larger than 64K, the whole pro
gram may"
370  PRINT "reside in memory, and the execution of the program will be speeded u
p."
400  INPUT "Does your machine have a memory larger than 64K (Y/N)?",B$
410  IF (UPC$(B$)<>"Y") AND (UPC$(B$)<>"N") THEN 400
420  IF UPC$(B$)="Y" THEN Overlay=0
430  IF UPC$(B$)="N" THEN Overlay=1
440  PRINT PAGE;"Please wait a few minutes while this program writes some other"
450  PRINT "programs which will help handle your data."
460  ASSIGN #3 TO A$
470  File$=A$
480  ASSIGN #5 TO "DBTMP1",Check
490  BUFFER #1
500  BUFFER #3
510  IF Check<>0 THEN 530
520  PURGE "DBTMP1"
530  CREATE "DBTMP1",20
540  ASSIGN #5 TO "DBTMP1"
550  BUFFER #5
560  PRINT #5;"10    SUB Entry(#9,Recno,Key,Key$,Keyflag)"
570  PRINT #5;"20    ! This is a program-written program to enter data from the"
580  PRINT #5;"30    !  keyboard."
590  PRINT #5;"40    !  #9 is the data file itself"
600  PRINT #5;"50    !  Recno is the record number reserved for the data"
610  PRINT #5;"60    !  Key -- used for the key if it is numeric"
620  PRINT #5;"70    !  Key$ -- used for the key if it is a string"
630  PRINT #5;"80    !  Keyflag -- 1 if key is numeric, 0 if key is a string"
640  I=90
650  GOSUB Setb
660  READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
670  IF Keytype=2 THEN Keytype=0
680  Keyflag=Keytype
690  READ #3,D
700  READ #3;A$
710  IF POS(A$,"Key")=0 THEN 740
720  Keydim$=A$
730  GOTO Donedim
740  B$[6]=A$
750  PRINT #5;B$
```

```
760  GOSUB Setb
770  GOTO 700


******** THIS SECTION CONSTRUCTS THE FOR-NEXT LOOPS **
******


790 Donedim:  READ #3,A  ! Position the pointer to where the titles ar
e
800  PRINT #5;VAL$(I)&"PRINT PAGE;"&CHR$(34)&"As the title of each fie
ld is displayed with its type"&CHR$(34)
810  I=I+10
820  PRINT #5;VAL$(I)&"PRINT "&CHR$(34)&"(numeric or string), please e
nter the proper value."&CHR$(34)&",LIN(2)"
830  I=I+10
840  I$="INPUT "&CHR$(34)&"Please enter the value for "
850  D$="DISP "&CHR$(34)&"Please enter the value for "
860  Nonums=Nostrs=Nonummats=Nostrmats=0
870 Readnextformat: READ #1;X
880                 IF X=0 THEN Doneformat
890  ON X GOTO Simplenumber,Simplestring,Matrixofnumbers,Matrixofstrin
gs
900 Simplenumber:  READ #3;Title$
910       Nonums=Nonums+1
920       GOSUB Setb
930       B$[6]=I$&Title$&"  (numeric)"&CHR$(34)&",N"&VAL$(Nonums)
940       PRINT #5;B$
950       GOSUB Setb
960       B$[6]="PRINT "&CHR$(34)&Title$&":"&CHR$(34)&";N"&VAL$(Nonums
)
970       PRINT #5;B$
980       GOTO Readnextformat
990 Simplestring:  READ #3;Title$
1000      READ #1;X          ! Read past string length
1010      Nostrs=Nostrs+1
1020      GOSUB Setb
1030      B$[6]="INPUT "&I$&Title$&"  (string)"&CHR$(34)&",S"&VAL$(Nostrs)&
"$"
1040      PRINT #5;B$
1050      GOSUB Setb
1060      B$[6]="PRINT "&CHR$(34)&Title$&":"&CHR$(34)&";S"&VAL$(Nostrs
)&"$"
1070      PRINT #5;B$
1080      GOTO Readnextformat
1090 Matrixofnumbers:  READ #3;Title$
1100      !Nonummats=Nonummats+1
```

```
1110       READ #1;X       ! Read the dimensionality of the array
1120       FOR J=1 TO X    ! Read the dimension bounds of the array
1130          READ #1;S(J,1),S(J,2)
1140       NEXT J
1150       Subscripts$="("
1160       Subprompts$="("&CHR$(34)&";"
1170       FOR J=1 TO X
1180       Subscripts$=Subscripts$&"I"&VAL$(J)&","
1190       Subprompts$=Subprompts$&"I"&VAL$(J)&";"&CHR$(34)&","&CHR$(34
)&";"
1200       PRINT #5;VAL$(I)&"        FOR I"&VAL$(J)&"="&VAL$(S(J,1))&" T
O "&VAL$(S(J,2))
1210       I=I+10
1220       NEXT J
1230       Subscripts$[LEN(Subscripts$)]=")"
1240       Subprompts$[LEN(Subprompts$)-2]=")"
1250       GOSUB Setb
1260       B$[6]=D$&Title$&Subprompts$&"  (numeric)"&CHR$(34)&";"
1270       PRINT #5;B$
1280       GOSUB Setb
1290       B$[6]="INPUT "&CHR$(34)&CHR$(34)&",Na"&VAL$(Nonummats)&Subsc
ripts$
1300       PRINT #5;B$
1310       GOSUB Setb
1320       B$[6]="PRINT "&CHR$(34)&Title$&Subprompts$&":"&CHR$(34)&";Na
"&VAL$(Nonummats)&Subscripts$
1330       PRINT #5;B$
1340       FOR J=X TO 1 STEP -1
1350       PRINT #5;VAL$(I)&"        NEXT I"&VAL$(J)
1360       I=I+10
1370       NEXT J
1380       GOTO Readnextformat
1390 Matrixofstrings:  READ #3;Title$
1400       Nostrmats=Nostrmats+1
1410       READ #1;L,X
1420       FOR J=1 TO X    ! Read the dimension bounds of the array
1430          READ #1;S(J,1),S(J,2)
1440       NEXT J
1450       Subscripts$="("
1460       Subprompts$="("&CHR$(34)&";"
1470       FOR J=1 TO X
1480       Subscripts$=Subscripts$&"I"&VAL$(J)&","
1490       Subprompts$=Subprompts$&"I"&VAL$(J)&";"&CHR$(34)&","&CHR$(34
)&";"
1500       PRINT #5;VAL$(I)&"        FOR I"&VAL$(J)&"="&VAL$(S(J,1))&" T
O "&VAL$(S(J,2))
1510       I=I+10
```

```
1520       NEXT J
1530       Subscripts$[LEN(Subscripts$)]=")"
1540       Subprompts$[LEN(Subprompts$)-2]=")"
1550       GOSUB Setb
1560       B$[6]=D$&Title$&Subprompts$&"  (string)"&CHR$(34)&";"
1570       PRINT #5;B$
1580       GOSUB Setb
1590       B$[6]="LINPUT "&CHR$(34)&CHR$(34)&",Sa"&VAL$(Nostrmats)&"$"&
Subscripts$
1600       PRINT #5;B$
1610       GOSUB Setb
1620       B$[6]="PRINT "&CHR$(34)&Title$&Subprompts$&":"&CHR$(34)&";Sa
"&VAL$(Nostrmats)&"$"&Subscripts$
1630       PRINT #5;B$
1640       FOR J=X TO 1 STEP -1
1650       PRINT #5;VAL$(I)&"        NEXT I"&VAL$(J)
1660       I=I+10
1670       NEXT J
1680       GOTO Readnextformat
1690 Doneformat: READ #3,B   ! Get ready to read the PRINT # statement
s
1700       GOSUB Setb
1710       B$[6]="READ #9,Recno"
1720       PRINT #5;B$
1730       X=TYP(-3)
1740       IF X=3 THEN Doneprint
1750       READ #3;A$
1760       B$=VAL$(I)&"     "
1770       B$[6]=A$
1780       PRINT #5;B$
1790       I=I+10
1800       GOTO 1730
1810 Doneprint: READ #3,E;A$  ! Read the key assignment statement
1820       GOSUB Setb
1830       B$[6]=A$
1840       PRINT #5;B$
1850       PRINT #5;VAL$(I)&" SUBEXIT"
1860 ASSIGN #5 TO "DBTMP2",Check
1870 IF Check<>0 THEN 1890
1880 PURGE "DBTMP2"
1890 CREATE "DBTMP2",40
1900 ASSIGN #5 TO "DBTMP2"
1910 BUFFER #5
1920 PRINT #5;"10   SUB Edit(#9,Recno,Key,Key$,Keyflag)"
1930 PRINT #5;"20   ! This is a program-written program to edit data."
1940 PRINT #5;"30   !  #9 is the data file itself"
1950 PRINT #5;"40   !  Recno is the record number of the data"
```

```
1960 PRINT #5;"50    !  Key and Key$ will hold the key "
1970 PRINT #5;"60    !  Keyflag=1 -> numeric key, Keyflag=0 -> string k
ey"
1980 S$="     "
1990 I=70
2000 READ #1,D                    ! Get the DIM statements
2010 READ #1;A$
2020 IF POS(A$,"Key")<>0 THEN 2070
2030 GOSUB Setb
2040 B$[6]=A$
2050 PRINT #5;B$
2060 GOTO 2010
2070 READ #1,C                    ! Get the READ# statements
2080 GOSUB Setb
2090 B$[6]="READ #9,Recno         ! Position the pointer "
2100 PRINT #5;B$
2110 IF TYP(-1)=3 THEN Doneread1
2120 READ #1;A$
2130 GOSUB Setb
2140 B$[6]=A$
2150 PRINT #5;B$
2160 GOTO 2110
2170 Doneread1: GOSUB Setb
2180            B$[6]="PRINT PAGE;"&CHR$(34)&"The title of each field a
nd its previous value(s)"&CHR$(34)
2190            PRINT #5;B$
2200            GOSUB Setb
2210            B$[6]="PRINT "&CHR$(34)&"will be displayed on the scree
n.  You will then be"&CHR$(34)
2220            PRINT #5;B$
2230            GOSUB Setb
2240            B$[6]="PRINT "&CHR$(34)&"asked whether or not you wish
to change the values."&CHR$(34)&",LIN(2)"
2250            PRINT #5;B$
2260            READ #1,1
2270            READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
 ! Position pointer at formats
2280            READ #3,A    ! Position pointer at titles
2290            Nonums=Nostrs=Nonummats=Nostrmats=0
2300 Readnext:  READ #1;X
2310            IF X=0 THEN Donefor
2320            READ #3;Title$
2330            ON X GOTO Number,String,Matno,Matstr
2340 Number:    Nonums=Nonums+1
2350            GOSUB Setb
2360            B$[6]="PRINT "&CHR$(34)&"OLD---"&Title$&":"&CHR$(34)&";
N"&VAL$(Nonums)
```

```
2370            PRINT #5;B$
2380            GOSUB Setb
2390            B$[6]="DISP "&CHR$(34)&"Enter a new value, or press CON
T to keep the old one for "&Title$&CHR$(34)&";"
2400            PRINT #5;B$
2410            GOSUB Setb
2420            B$[6]="INPUT "&CHR$(34)&CHR$(34)&",N"&VAL$(Nonums)
2430            PRINT #5;B$
2440            GOSUB Setb
2450            B$[6]="PRINT "&CHR$(34)&"NEW---"&Title$&":"&CHR$(34)&";
N"&VAL$(Nonums)
2460            PRINT #5;B$
2470            GOTO Readnext
2480 String:    Nostrs=Nostrs+1
2490            READ #1;X     ! Read past the string length
2500            GOSUB Setb
2510            B$[6]="PRINT "&CHR$(34)&"OLD---"&Title$&": "&CHR$(34)&"
;S"&VAL$(Nostrs)&"$"
2520            PRINT #5;B$
2530            GOSUB Setb
2540            B$[6]="DISP "&CHR$(34)&"Enter a new value, or press CON
T to keep the old one for "&Title$&CHR$(34)&";"
2550            PRINT #5;B$
2560            GOSUB Setb
2570            B$[6]="EDIT "&CHR$(34)&CHR$(34)&",S"&VAL$(Nostrs)&"$"
2580            PRINT #5;B$
2590            GOSUB Setb
2600            B$[6]="PRINT "&CHR$(34)&"NEW---"&Title$&": "&CHR$(34)&"
;S"&VAL$(Nostrs)&"$"
2610            PRINT #5;B$
2620            GOTO Readnext
2630 Matno:     Nonummats=Nonummats+1
2640            H$=VAL$(Nonummats)
2650            Varnam$="Na"&H$
2660            Label$="m"
2670            Type$="  (numeric)"
2680            GOTO Common
2690 Matstr:    Nostrmats=Nostrmats+1
2700            READ #1;L     ! Read past the string length
2710            H$=VAL$(Nostrmats)
2720            Varnam$="Sa"&H$&"$"
2730            Label$="s"
2740            Type$="  (string)"
2750 Common:    READ #1;X     ! Read the dimensionality of the array
2760            FOR J=1 TO X ! Read the dimensions of the array
2770               READ #1;S(J,1),S(J,2)
2780            NEXT J
```

```
******      Print the matrix      ******

2800            GOSUB Setb
2810            B$[6]="PRINT "&CHR$(34)&"OLD---"&Title$&":"&CHR$(34)
2820            PRINT #5;B$
2830            GOSUB Setb
2840            B$[6]="PRINT "&Varnam$&"(*);"
2850            PRINT #5;B$

******      Ask if everything is alright      ******

2870            GOSUB Setb
2880            B$[6]="Ask"&Label$&H$&": DISP "&CHR$(34)&"Do you wish t
o change anything under the array "&Title$&CHR$(34)&";"
2890            PRINT #5;B$
2900            GOSUB Setb
2910            B$[6]="INPUT "&CHR$(34)&" (Y/N)?"&CHR$(34)&",Answer$"
2920            PRINT #5;B$
2930            GOSUB Setb
2940            B$[6]="IF Answer$="&CHR$(34)&"Y"&CHR$(34)&" OR Answer$=
"&CHR$(34)&"y"&CHR$(34)&" THEN Edit"&Label$&H$
2950            PRINT #5;B$
2960            GOSUB Setb
2970            B$[6]="IF Answer$="&CHR$(34)&"N"&CHR$(34)&" OR Answer$=
"&CHR$(34)&"n"&CHR$(34)&" THEN Noedit"&Label$&H$
2980            PRINT #5;B$
2990            GOSUB Setb
3000            B$[6]="GOTO Ask"&Label$&H$
3010            PRINT #5;B$

******      Select Editing Mode      ******

3030            GOSUB Setb
3040            B$[6]="Edit"&Label$&H$&": PRINT LIN(2),"&CHR$(34)&"Ther
e are two modes of editing an array."&CHR$(34)
3050            PRINT #5;B$
3060            GOSUB Setb
3070            B$[6]="PRINT "&CHR$(34)&"Mode 1 allows element-by-eleme
nt editing of the array"&CHR$(34)
3080            PRINT #5;B$
```

```
3090            GOSUB Setb
3100            B$[6]="PRINT "&CHR$(34)&"Mode 2 allows you to specify t
he subscript of the element you wish"&CHR$(34)
3110            PRINT #5;B$
3120            GOSUB Setb
3130            B$[6]="PRINT "&CHR$(34)&"       to change"&CHR$(34)&",L
IN(2)"
3140            PRINT #5;B$
3150            GOSUB Setb
3160            B$[6]="Askmode"&Label$&H$&": INPUT "&CHR$(34)&"Please e
nter the mode you wish to use (1 or 2)"&CHR$(34)&",Mode"
3170            PRINT #5;B$
3180            GOSUB Setb
3190            B$[6]="Mode=INT(Mode)"
3200            PRINT #5;B$
3210            GOSUB Setb
3220            B$[6]="IF Mode<1 OR Mode>2 THEN Askmode"&Label$&H$
3230            PRINT #5;B$
3240            GOSUB Setb
3250            B$[6]="ON Mode GOTO All"&Label$&H$&",Single"&Label$&H$
3260            PRINT #5;B$


******      Single element editing mode      ******


3280            GOSUB Setb
3290            B$[6]="Single"&Label$&H$&": PRINT "&CHR$(34)&"Please en
ter the subscripts of the element you wish to change,"&CHR$(34)
3300            PRINT #5;B$
3310            GOSUB Setb
3320            B$[6]="PRINT "&CHR$(34)&"one at a time.  The valid rang
es for each subscript will be"&CHR$(34)
3330            PRINT #5;B$
3340            GOSUB Setb
3350            B$[6]="PRINT "&CHR$(34)&"displayed as they are needed.
(This array has "&VAL$(X)&" dimension(s).)"&CHR$(34)&",LIN(2)"
3360            PRINT #5;B$
3370            Subscripts$="("
3380            Subprompts$="("&CHR$(34)&";"
3390            FOR J=1 TO X        ! Enter the subscripts of the elemen
t to be
3400            J$=VAL$(J)          !  changed
3410            GOSUB Setb
3420            B$[6]="Su"&Label$&H$&J$&": DISP "&CHR$(34)&"Dimension #
"&J$&" (Range: "&VAL$(S(J,1))&" TO "&VAL$(S(J,2))&")"&CHR$(34)&";"
3430            PRINT #5;B$
```

```
3440            GOSUB Setb
3450            B$[6]="INPUT I"&J$
3460            PRINT #5;B$
3470            GOSUB Setb
3480            B$[6]="I"&J$&"=INT(I"&J$&")"
3490            PRINT #5;B$
3500            GOSUB Setb
3510            B$[6]="IF I"&J$&"<"&VAL$(S(J,1))&" OR I"&J$&">"&VAL$(S(
J,2))&" THEN Su"&Label$&H$&J$       ! Check to be sure the subscript fa
lls within range
3520            PRINT #5;B$
3530            Subscripts$=Subscripts$&"I"&J$&","
3540            Subprompts$=Subprompts$&"I"&J$&";"&CHR$(34)&","&CHR$(34
)&";"
3550            NEXT J
3560            Subscripts$[LEN(Subscripts$)]=")"
3570            Subprompts$[LEN(Subprompts$)-2]=")"
3580            GOSUB Setb
3590            B$[6]=D$&Title$&Subprompts$&Type$&CHR$(34)&";"
3600            PRINT #5;B$
3610            GOSUB Setb
3620            B$[6]="INPUT "&CHR$(34)&CHR$(34)&","&Varnam$&Subscripts
$
3630            PRINT #5;B$
3640            GOSUB Setb
3650            B$[6]="PRINT "&CHR$(34)&"NEW---"&Title$&Subprompts$&":"
&CHR$(34)&";"&Varnam$&Subscripts$
3660            PRINT #5;B$
3670            GOSUB Setb
3680            B$[6]="GOTO Ask"&Label$&H$
3690            PRINT #5;B$


******      All elements editing mode      ******


3710            GOSUB Setb
3720            B$[6]="All"&Label$&H$&": PRINT LIN(2),"&CHR$(34)&"As th
e prompt for each element of the array is"&CHR$(34)
3730            PRINT #5;B$
3740            GOSUB Setb
3750            B$[6]="PRINT "&CHR$(34)&"displayed, you may either ente
r a new value for the element,"&CHR$(34)
3760            PRINT #5;B$
3770            GOSUB Setb
3780            B$[6]="PRINT "&CHR$(34)&"or just press CONT to retain t
he old value."&CHR$(34)&",LIN(2)"
```

```
3790            PRINT #5;B$
3800            Subscripts$="("
3810            Subprompts$="("&CHR$(34)&";"
3820            FOR J=1 TO X
3830            J$=VAL$(J)
3840            Subscripts$=Subscripts$&"I"&J$&","
3850            Subprompts$=Subprompts$&"I"&J$&";"&CHR$(34)&","&CHR$(34
)&";"
3860            GOSUB Setb
3870            B$[8]="FOR I"&J$&"="&VAL$(S(J,1))&" TO "&VAL$(S(J,2))
3880            PRINT #5;B$
3890            NEXT J
3900            Subscripts$[LEN(Subscripts$)]=")"
3910            Subprompts$[LEN(Subprompts$)-2]=")"
3920            GOSUB Setb
3930            B$[6]=D$&Title$&Subprompts$&Type$&CHR$(34)&";"&Varnam$&
Subscripts$&";"
3940            PRINT #5;B$
3950            GOSUB Setb
3960            B$[6]="INPUT "&CHR$(34)&CHR$(34)&","&Varnam$&Subscripts
$
3970            PRINT #5;B$
3980            GOSUB Setb
3990            B$[6]="PRINT "&CHR$(34)&"NEW---"&Title$&Subprompts$&":"
&CHR$(34)&";"&Varnam$&Subscripts$
4000            PRINT #5;B$
4010            FOR J=X TO 1 STEP -1
4020            GOSUB Setb
4030            B$[8]="NEXT I"&VAL$(J)
4040            PRINT #5;B$
4050            NEXT J
4060            GOSUB Setb
4070            B$[6]="GOTO Ask"&Label$&H$
4080            PRINT #5;B$


******      Done editing      ******


4100            GOSUB Setb
4110            B$[6]="Noedit"&Label$&H$&": !"
4120            PRINT #5;B$
4130            GOTO Readnext
```

```
******      Re-store the record      ******

4150 Donefor:  READ #3,B
4160           GOSUB Setb
4170           B$[6]="READ #9,Recno"
4180           PRINT #5;B$
4190           READ #3;A$
4200           GOSUB Setb
4210           B$[6]=A$
4220           PRINT #5;B$
4230           IF TYP(-3)<>3 THEN 4190
4240           READ #1,E;A$
4250           GOSUB Setb
4260           B$[6]=A$
4270           PRINT #5;B$
4280           GOSUB Setb
4290           B$[6]="SUBEXIT"
4300           PRINT #5;B$
4310 ASSIGN #5 TO "DBTMP4",Check
4320 IF Check<>0 THEN 4340
4330 PURGE "DBTMP4"
4340 CREATE "DBTMP4",20
4350 ASSIGN #5 TO "DBTMP4"
4360 BUFFER #5
4370 PRINT #5;"10   SUB Replacekey(#9,Recno,Key,Key$,Keyflag)"
4380 PRINT #5;"20   ! This is a program-written program that allows th
e user"
4390 PRINT #5;"30   !  to replace only the key of his data record in t
he event"
4400 PRINT #5;"40   !  that he has tried to assign a duplicate key to
it."
4410 PRINT #5;"50   !  #9 is the data file itself"
4420 PRINT #5;"60   !  Recno is the record number of the data"
4430 PRINT #5;"70   !  Key or Key$ is the faulty key"
4440 PRINT #5;"80   !  Keyflag=0 -> string key;  Keyflag=1 -> numeric.
"
4450 PRINT #5;"89   DIM Title$[30]"
4460 PRINT #5;"90   READ #9,1"
4470 PRINT #5;"91   READ #9;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Po
skey"
4480 PRINT #5;"100  READ #9,A    ! Read the titles"
4490 PRINT #5;"110  FOR I=1 TO Poskey"
4500 PRINT #5;"120     READ #9;Title$"
4510 PRINT #5;"130  NEXT I"
```

```
4520 PRINT #5;"135  BEEP"
4530 PRINT #5;"140  IF Keyflag THEN N"
4540 PRINT #5;"150 S: Type$="&CHR$(34)&"string"&CHR$(34)
4550 PRINT #5;"160  GOTO A"
4560 PRINT #5;"170 N: Type$="&CHR$(34)&"numeric"&CHR$(34)
4570 PRINT #5;"180 A: PRINT LIN(2),"&CHR$(34)&"The "&CHR$(34)&"&Type$&
"&CHR$(34)&" value you have entered for the key field of"&CHR$(34)
4580 PRINT #5;"190  PRINT "&CHR$(34)&"this data record has already bee
n used.  In order to avoid"&CHR$(34)
4590 PRINT #5;"200  PRINT "&CHR$(34)&"ambiguity, each data record must
 have a unique value in"&CHR$(34)
4600 PRINT #5;"210  PRINT "&CHR$(34)&"its key field (the title of your
 keyfield is "&CHR$(34)&"&Title$&"&CHR$(34)&")"&CHR$(34)
4610 I=220
4620 READ #1,D
4630 READ #1;A$
4640 IF POS(A$,"Key")<>0 THEN Getread
4650 GOSUB Setb
4660 B$[6]=A$
4670 PRINT #5;B$
4680 GOTO 4630
4690 Getread: READ #1,C
4700          GOSUB Setb
4710          B$[6]="READ #9,Recno"
4720          PRINT #5;B$
4730          IF TYP(-1)=3 THEN Doneread
4740          READ #1;A$
4750          GOSUB Setb
4760          B$[6]=A$
4770          PRINT #5;B$
4780          GOTO 4730
4790 Doneread:READ #1,E;C$     ! Read the key assignment
4800          Pes=POS(C$,"=")
4810          Pbl=POS(C$," ")
4820          A$=C$[Pes+1,Pbl-1]
4830          Plp=POS(A$,"(")
4840          Prp=POS(A$,")")
4850          GOSUB Setb
4860          B$[6]="DISP "&CHR$(34)&"Please enter a new "&CHR$(34)&"&
Type$&"&CHR$(34)&" value for "&CHR$(34)&"&Title$"
4870          IF Plp<>0 THEN B$=B$&"&"&CHR$(34)&A$[Plp,Prp]&CHR$(34)
4880          B$=B$&";"
4890          PRINT #5;B$
4900          GOSUB Setb
4910          B$[6]="INPUT "&CHR$(34)&CHR$(34)&","&C$[1,Pes-1]
4920          PRINT #5;B$
4930          GOSUB Setb
```

```
4940          B$[6]="DISP "&CHR$(34)&"Thank you"&CHR$(34)
4950          PRINT #5;B$
4960          GOSUB Setb
4970          B$[6]="WAIT 500"
4980          PRINT #5;B$
4990          GOSUB Setb
5000          B$[6]=A$&"="&C$[1,Pes-1]
5010          PRINT #5;B$
5020          READ #1,B
5030          GOSUB Setb
5040          B$[6]="READ #9,Recno"
5050          PRINT #5;B$
5060          READ #1;A$
5070          GOSUB Setb
5080          B$[6]=A$
5090          PRINT #5;B$
5100          IF TYP(-1)<>3 THEN 5060
5110          GOSUB Setb
5120          B$[6]="SUBEXIT"
5130          PRINT #5;B$
5140 ASSIGN #5 TO "DBTMP3",Check
5150 IF Check<>0 THEN 5170
5160 PURGE "DBTMP3"
5170 CREATE "DBTMP3",5
5180 ASSIGN #5 TO "DBTMP3"
5190 BUFFER #5
5200 I=10
5210 GOSUB Setb
5220 B$[6]="OPTION BASE 1"
5230 PRINT #5;B$
5240 GOSUB Setb
5250 B$[6]=Keydim$
5260 PRINT #5;B$
5270 GOSUB Setb
5280 B$[6]="Keyflag="&VAL$(Keytype)
5290 PRINT #5;B$
5300 GOSUB Setb
5310 B$[6]="ASSIGN#1 TO "&CHR$(34)&File$&CHR$(34)
5320 PRINT #5;B$
5330 GOSUB Setb
5340 B$[6]="ASSIGN#3 TO "&CHR$(34)&File$&CHR$(34)
5350 PRINT #5;B$
5360 GOSUB Setb
5370 B$[6]="Keylength="&VAL$(Keylength)
5380 PRINT #5;B$
5390 GOSUB Setb
5400 B$[6]="Maxsize="&VAL$(Maxsize)
```

```
5410 PRINT #5;B$
5420 GOSUB Setb
5430 B$[6]="READ #1,"&VAL$(F)&";Dirname$"
5440 PRINT #5;B$
5450 GOSUB Setb
5460 B$[6]="READ #1,"&VAL$(G)&";X,Keysinmemory"
5470 PRINT #5;B$
5480 GOSUB Setb
5490 B$[6]="Device$="&CHR$(34)&Device$&CHR$(34)
5500 PRINT #5;B$
5510 GOSUB Setb
5520 B$[6]="LINK "&CHR$(34)&"DBDRVR"&CHR$(34)&"&Device$,200"
5530 PRINT #5;B$
5540 GOSUB Setb
5550 B$[6]="Overlay="&VAL$(Overlay)
5560 PRINT #5;B$
5570 GET "DBTMP3",10,10
5580 Setb: B$=VAL$(I)&S$
5590       I=I+2
5600       RETURN
```

# Listing of File "DBDRVR"

```
200  ASSIGN #2 TO Dirname$
204  Dirnorecs=(INT(Maxsize/Keysinmemory)+1)*2
208  IF Overlay THEN 276
212  LINK "DBREPO"&Device$,1000
216  LINK "DBLTKY"&Device$,2200
220  LINK "DBSUB1"&Device$,2350
244  LINK "DBEXSS"&Device$,3950
256  LINK "DBEXSN"&Device$,4450
264  LINK "DBTMP1",6000
268  LINK "DBTMP2",7000
272  LINK "DBTMP4",8000
276  Cnumber=0       ! Initialize command flag
280  PRINT PAGE;"   Please type in one of the valid commands (listed b
elow),"
284  PRINT "and press CONT."
288  PRINT LIN(2),"VALID COMMANDS:"
292  PRINT SPA(5);"ENTER";TAB(18);"Enter a data record from the keyboa
rd"
296  PRINT SPA(5);"LIST KEYS   List the values of the key field of eac
h record in"
300  PRINT SPA(17);"the data base"
304  PRINT SPA(5);"EDIT";TAB(18);"Edit a data record from the keyboard
"
308  PRINT SPA(5);"EDIT LAST   Edit the data record whose key was last
 referenced"
312  PRINT SPA(5);"DELETE";TAB(18);"Delete a record"
316  PRINT SPA(5);"REPORT";TAB(18);"Report on the data base"
320  PRINT SPA(5);"STOP";TAB(18);"End the program"
324  GOTO 340
328 Command:  BEEP
332           DISP Command$;" is an invalid command.  Please try again
."
336           WAIT 1000
340 Take:  INPUT "Command?",Command$
344  Command$=UPC$(Command$)
348  IF Command$="STOP" THEN Stop
352  IF Command$="ENTER" THEN Enter
356  IF Command$="LIST KEYS" THEN Listkeys
360  IF Command$="EDIT" THEN Edit
364  IF Command$="EDIT LAST" THEN Editlast
368  IF Command$="REPORT" THEN Report
372  IF Command$="DELETE" THEN Delete
376  GOTO Command
380 Stop:  PRINT PAGE;"PROGRAM TERMINATED"
384        DISP ""
388        BEEP
392        END
```

```
396 Enter: IF NOT Overlay THEN 408
400        IF (Cnumber=1) OR (Cnumber=2) OR (Cnumber=3) OR (Cnumber=4)
 THEN 408
404        GOSUB Link1
408        Cnumber=1
412        CALL Findemptyrecord(#1,Recno)
416        IF Recno THEN 440
420        BEEP
424        PRINT LIN(2),"FILE IS FULL. THERE IS NO MORE ROOM."
428        DISP "FILE IS FULL.  THERE IS NO MORE ROOM."
432        WAIT 700
436        GOTO 288
440        CALL Entry(#1,Recno,Key,Key$,Keyflag)
444        CALL Hashpage(Pageno,Key,Key$,Dirnorecs,Keyflag)
448        CALL Updatepage(#2,Pageno,Key,Key$,Keylength,Recno,Keyflag,
Keysinmemory,Dirnorecs,Collide)
452        IF NOT Collide THEN 288
456        Collide=0
460        CALL Replacekey(#1,Recno,Key,Key$,Keyflag)
464        GOTO 444
468 Edit:  IF NOT Overlay THEN 480
472        IF (Cnumber=1) OR (Cnumber=2) OR (Cnumber=3) OR (Cnumber=4)
 THEN 480
476        GOSUB Link1
480        Cnumber=2
484        CALL Getkey(#1,Key,Key$,Keyflag,"edit")
488        GOTO 508
492 Editlast: IF NOT Overlay THEN 504
496        IF (Cnumber=1) OR (Cnumber=2) OR (Cnumber=3) OR (Cnumber=4)
 THEN 504
500        GOSUB Link1
504        Cnumber=3
508        CALL Hashpage(Pageno,Key,Key$,Dirnorecs,Keyflag)
512        CALL Findkey(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysin
memory,Dirnorecs,Maxsize,1)
516        GOTO 288
520 Delete:IF NOT Overlay THEN 532
524        IF (Cnumber=1) OR (Cnumber=2) OR (Cnumber=3) OR (Cnumber=4)
 THEN 532
528        GOSUB Link1
532        Cnumber=4
536        CALL Getkey(#1,Key,Key$,Keyflag,"delete")
540        CALL Hashpage(Pageno,Key,Key$,Dirnorecs,Keyflag)
544        CALL Findkey(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysin
memory,Dirnorecs,Maxsize,0)
548        GOTO 288
552 Listkeys: IF NOT Overlay THEN 564
```

```
556         IF (Cnumber=5) OR (Cnumber=6) THEN 564
560         GOSUB Link2
564         Cnumber=5
568         PRINT LIN(2),"KEYS:"
572         CALL Listkeys(#2,Dirnorecs,Keysinmemory,Keyflag,Keylength)
576         PRINT LIN(2)
580         GOTO 288
584 Report: A$=""
592         INPUT "Do you want to use an old report format (Y/N)?",A$
596         IF UPC$(A$)="N" THEN 680
600         IF UPC$(A$)="Y" THEN 612
604         BEEP
608         GOTO 584
612         IF NOT Overlay OR (Cnumber=5) OR (Cnumber=6) THEN 620
616         GOSUB Link2
617         Cnumber=6
620         INPUT "Please enter the name of the file of the old report"
,Reportfile$
624         IF LEN(Reportfile$)<=6 THEN 644
628         BEEP
632         DISP "IMPROPER FILE NAME"
636         WAIT 1000
640         GOTO 620
644         ASSIGN #10 TO Reportfile$,C
648         IF NOT C THEN 668
652         BEEP
656         DISP "ILLEGAL FILE"
660         WAIT 1000
664         GOTO 620
668         LINK Reportfile$,9000
672         Already_saved=1
676         GOTO 712
680         IF NOT Overlay THEN 690
684         DISP "PLEASE WAIT WHILE THE REPORT WRITER IS PAGED IN"
688         LINK "DBREPO"&Device$,1000
689         Cnumber=6
690         CALL Sortwrite(#1,Sorttype,Sortlength,Keyflag)
692         CALL Report1(#1,#2,#3,Sorttype,Sortlength)
696         Already_saved=0
700         IF NOT Overlay THEN 708
704         GOSUB Link2
708         LINK "DBTMP5",9000
709         LINK "DBTMP6",9810
712         CALL Merge(#1,#2,#5,Maxsize,Dirnorecs,Keylength,Keysinmemor
y)
732         CALL Report(#1,#5)
736         IF Already_saved THEN 288
```

```
740         BEEP
744         A$=""
748         INPUT "Would you like to save this report format (Y/N)?",A$
752         IF UPC$(A$)="N" THEN 288
756         IF UPC$(A$)="Y" THEN 768
760         BEEP
764         GOTO 744
768         INPUT "What would you like to name the file (up to six char
acters)?",Reportfile$
772         IF LEN(Reportfile$)<=6 THEN 792
776         BEEP
780         DISP "ILLEGAL FILE NAME"
784         WAIT 1000
788         GOTO 768
792         ASSIGN #10 TO Reportfile$,C
796         IF C=1 THEN 816
800         BEEP
804         DISP "File already exists.  Please pick a different one."
808         WAIT 1000
812         GOTO 768
816         SAVE Reportfile$,9000
820         GOTO 288
824 Link1:  DISP "PLEASE WAIT WHILE THE PROPER ROUTINES ARE PAGED IN"
828         LINK "DBSUB1"&Device$,1000
860         LINK "DBTMP1"&Device$,3000
864         LINK "DBTMP2"&Device$,4000
868         LINK "DBTMP4"&Device$,5000
872         RETURN
876 Link2:  DISP "PLEASE WAIT WHILE THE MERGE AND SORT ROUTINES ARE PAG
ED IN"
880         LINK "DBLTKY"&Device$,1000
888         LINK "DBEXSS"&Device$,2000
896         LINK "DBEXSN"&Device$,3000
904         RETURN
```

# Listing of File "DBREPO"

```
1000 SUB Report1(#1,#2,#3,Sorttype,Sortlength)
1002 DIM Title$[30]
1004 Device=-1
1006 INPUT "Do you want the report on an HPIB printer (Y/N)?",Answer$
1008 IF UPC$(Answer$)="Y" THEN Enterdevice
1010 IF UPC$(Answer$)="N" THEN Enterselect
1012 GOTO 1006
1014 Enterdevice: INPUT "Please enter the HPIB address of the printer"
,Device
1016 Enterselect: INPUT "Please enter the select code of the printer",
Select
1018 INPUT "How many columns per line does the printer have?",Colperli
ne
1020 Colperline=INT(Colperline)
1022 IF (Colperline>=40) AND (Colperline<=132) THEN 1032
1024 BEEP
1026 DISP "PRINTER WIDTH MUST BE BETWEEN 40 AND 132 COLUMNS"
1028 WAIT 1000
1030 GOTO 1018
1032 Titles:PRINT PAGE;"There are two ways to set up titles for the re
port."
1034         PRINT "1) One way is to print the title once at the very t
op of the  report."
1036         PRINT "   This would look nice for reports where each reco
rd only takes up"
1038         PRINT "   one or two lines."
1040         PRINT "2) The second way is to print the titles immediatel
y before every"
1042         PRINT "   record.  This would be the way to go if there ar
e any arrays in "
1044         PRINT "   the report."
1046         INPUT "Please enter 1 or 2 and press CONT to select the ti
tle mode you want",Titlemode
1048         Titlemode=INT(Titlemode)
1050         IF (Titlemode<1) OR (Titlemode>2) THEN 1046
1052 PRINT PAGE;"Below, the available fields (and their types) are lis
ted:"
1054 READ #1,1
1056 READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
1058 READ #3,A
1060 Count=0
1062 Readx: READ #1;X
1064         IF NOT X THEN Inputfields
1066         READ #3;Title$
1068         Count=Count+1
1070         Count$=VAL$(Count)&". "
1072         ON X GOTO Snum,Sstr,Anum,Astr
```

```
1074 Snum:  PRINT SPA(5);Count$&Title$&"-simple numeric"
1076        GOTO Readx
1078 Sstr:  READ #1;X
1080        PRINT SPA(5);Count$&Title$&"-simple string-length "&VAL$(X
)
1082        GOTO Readx
1084 Anum:  READ #1;X
1086        PRINT SPA(5);Count$&Title$&"-"&VAL$(X)&"-dimensional numer
ic array (";
1088        GOTO Readdim
1090 Astr:  READ #1;L,X
1092        PRINT SPA(5);Count$&Title$&"-"&VAL$(X)&"-dimensional strin
g array"
1094        PRINT SPA(LEN(Title$)+12);"(each element is "&VAL$(L)&" ch
aracters)"&"(";
1096 Readdim:FOR I=1 TO X
1098           READ #1;Y,Z
1100           PRINT VAL$(Y)&":"&VAL$(Z)&",";
1102        NEXT I
1104        PRINT )"
1106        GOTO Readx
1108 Inputfields:Howmanyfields=Count
1110        INPUT "How many of the above fields do you wish to have in
 your report?",Howmanyfields
1112        Howmanyfields=INT(Howmanyfields)
1114        IF (Howmanyfields<1) OR (Howmanyfields>Count) THEN Inputfi
elds
1116 CALL Report2(#1,#2,#3,Select,Device,Count,Titlemode,Howmanyfields
,Colperline,Sorttype,Sortlength)
1118 SUBEND
1120 SUB Report2(#1,#2,#3,Select,Device,Count,Titlemode,Howmanyfields,
Colperline,Sorttype,Sortlength)
1122 ! #1 and #3 are the data file
1124 !  #2 is the directory
1126 !  Select is the printer select code
1128 !  Device is the HPIB device address (-1 for non-HPIB printers)
1130 !  Count is the total number of fields in the data record
1132 !  Titlemode=1 for one set of titles; 2 for titles at every recor
d
1134 !  Howmanyfields for how many are on the report
1136 !  Colperline - the number of columns in the printer line
1138 OPTION BASE 1
1140 DIM Whichfields(Howmanyfields),S(6,2),Image$[160],Print$[160],A$[
160],B$[160],Imaget$[160],Printt$[160],Title$[30]
1144 Image$="Image1: IMAGE "
1146 Imaget$="Imaget1: IMAGE "
1148 Print$="PRINT USING Image1;"
```

```
1150 Printt$="PRINT USING Imaget1;"
1152 Titlecounter=1 ! Keep track of how many lines are needed for the
titles
1154 Imagecounter=1 ! Keep track of how many lines per record are need
ed (not
1156                ! counting the arrays in the record) for the reco
rd itself
1158 Usedflag=0      ! Usedflag tells whether or not the Print$ and Ima
ge# stings
1160                ! have been used.
1162 Roomleft=Colperline
1164 IF Howmanyfields<Count THEN 1174
1166 INPUT "Do you want to pick the order the fields will occur in (Y/
N)?",A$
1168 IF UPC$(A$)="Y" THEN 1174
1170 IF UPC$(A$)="N" THEN Auto
1172 GOTO 1166
1174 FOR I=1 TO Howmanyfields
1176    DISP "#"&VAL$(I)&":   ;
1178    INPUT "Please enter the title number of the next field",Whichf
ields(I)
1180    Whichfields(I)=INT(Whichfields(I))
1182    IF (Whichfields(I)>=1) OR (Whichfields(I)<=Count) THEN Next
1184    BEEP
1186    DISP "ILLEGAL"
1188    WAIT 500
1190    GOTO 1176
1192 Next: NEXT I
1194    GOTO After
1196 Auto:FOR I=1 TO Howmanyfields
1198    Whichfields(I)=I
1200    NEXT I
1202 After:ASSIGN #5 TO "DBTMP5",Check
1204    DISP "PLEASE WAIT FOR THE REPORT PROGRAM TO BE WRITTEN"
1206    IF Check THEN 1210
1208    PURGE "DBTMP5"
1210    CREATE "DBTMP5",20
1212    ASSIGN #5 TO "DBTMP5"
1214    PRINT #5;"10   SUB Report(#9,#2)"
1216    PRINT #5;"20   OPTION BASE 1"
1218    IF Device<>-1 THEN 1224
1220    PRINT #5;"30   PRINTER IS "&VAL$(Select)&",WIDTH("&VAL$(Colper
line)&")"
1222    GOTO 1226
1224    PRINT #5;"30   PRINTER IS "&VAL$(Select)&","&VAL$(Device)&",WI
DTH("&VAL$(Colperline)&")"
1226    PRINT #5;"40   Titlemode="&VAL$(Titlemode)
```

```
1228    READ #1,1
1230    READ #1;G,A,B,C,D
1232    READ #1,D       ! Set the file pointer to read the DIM stateme
nts
1234    Linenumber=50
1236    Tlinestart=201
1238    READ #1;A$      ! Read in the DIM statements
1240    GOSUB Setb
1242    B$[6]=A$
1244    PRINT #5;B$
1246    IF TYP(-1)<>3 THEN 1238! Check for the last one
1248    GOSUB Setb
1250    B$[6]="DIM Sort$["&VAL$(Sortlength)&"]"
1252    PRINT #5;B$
1254    GOSUB Setb
1256    B$[6]="ON END #2 GOTO Done"
1258    PRINT #5;B$
1260    GOSUB Setb
1262    B$[6]="READ #2,1     ! Set the pointer"
1264    PRINT #5;B$
1266    GOSUB Setb
1268    B$[6]="IF Titlemode-1 THEN Readnextpoint"
1270    PRINT #5;B$
1272    GOSUB Setb
1274    B$[6]="GOSUB 200"
1276    PRINT #5;B$
1278    GOSUB Setb
1280    ON Sorttype GOTO 1286,1282
1282    B$[6]="Readnextpoint: READ #2;Sort$,Pointer"
1284    GOTO 1288
1286    B$[6]="Readnextpoint: READ #2;Sort,Pointer"
1288    PRINT #5;B$
1290    GOSUB Setb
1292    B$[6]="READ #9,Pointer"
1294    PRINT #5;B$
1296    READ #1,C       ! Set the pointer at the READ # statements
1298    READ #1;A$      ! Read in the READ # statments
1300    GOSUB Setb
1302    B$[6]=A$
1304    PRINT #5;B$     ! Store the READ # statments
1306    IF TYP(-1)<>3 THEN 1298 ! Check for the last one
1308    GOSUB Setb
1310    B$[6]="IF Titlemode-2 THEN Skiptitles"
1312    PRINT #5;B$
1314    GOSUB Setb
1316    B$[6]="GOSUB 200"     ! Allow for two titlemodes
1318    PRINT #5;B$
```

```
1320     GOSUB Setb
1322     B$[6]="Skiptitles: !"
1324     PRINT #5;B$
1326     GOTO 1370


 ***  READ INFO ABOUT NEXT FIELD  ***


1330 Readj: FOR J=1 TO Whichfields(I)
1332         READ #3;Title$
1334         READ #1;X
1336         ON X GOTO N1,S1,Na1,Sa1
1338 N1:    Nonums=Nonums+1
1340         GOTO Nextj
1342 S1:    READ #1;L
1344         Nostrs=Nostrs+1
1346         GOTO Nextj
1348 Na1:   READ #1;Y
1350         Nonummats=Nonummats+1
1352         GOTO R1
1354 Sa1:   READ #1;L,Y
1356         Nostrmats=Nostrmats+1
1358 R1:    FOR K=1 TO Y
1360            READ #1;S(K,1),S(K,2)
1362         NEXT K
1364 Nextj:NEXT J
1366         RETURN


 ***  CHECK FOR NUMERIC AND STRING FIELDS WHICH WILL F
IT ON ONE LINE  ***


1370     FOR I=1 TO Howmanyfields
1372     Nonums=Nostrs=Nonummats=Nostrmats=Placesbefore=Placesafter=0
1374     READ #1,1
1376     READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
1378     READ #3,A
1380     GOSUB Readj
1382        ON X GOTO N2,S2,Na2,Sa2
1384 N2:    PRINT LIN(2),Title$&" is a numeric field.  You may select t
he format you wish"
1386        PRINT "to use on the report for this field.  Below, please
enter 1 if you wish"
1388        PRINT "to use fixed point format (i.e., 4.378, 9276.43, 166
79) or 2 if you wish"
```

```
1390        PRINT "to use floating point format (i.e., 3.799634E-92, 4E
+17, 3.2E+00)."
1392        INPUT "Please enter the format code (1 or 2)",Formatcode
1394        Formatcode=INT(Formatcode)
1396        IF Formatcode=1 THEN Fixedfor
1398        IF Formatcode=2 THEN Floatfor
1400        BEEP
1402        DISP "ILLEGAL FORMAT CODE!"
1404        WAIT 700
1406        GOTO 1392
1408 Fixedfor:INPUT "How many places before the decimal point should b
e allowed for ?",Placesbefore
1410        IF Placesbefore<0 THEN No
1412        IF Placesbefore<=12 THEN Floatfor
1414        BEEP
1416        DISP "MAXIMUM NUMBER OF PLACES BEFORE THE DECIMAL POINT IS
12"
1418        WAIT 1000
1420        GOTO 1392
1422 No:    BEEP
1424        DISP "NEGATIVE NUMBERS ARE ILLEGAL"
1426        WAIT 1000
1428        GOTO 1392
1430 Floatfor: INPUT "How many places after the decimal point should b
e allowed for?",Placesafter
1432        IF Placesafter<0 THEN No
1434        IF Placesafter<=12 THEN Okay
1436        BEEP
1438        DISP "MAXIMUM NUMBER OF PLACES AFTER THE DECIMAL POINT IS 1
2"
1440        WAIT 1000
1442        GOTO Floatfor
1444 Okay:  ON Formatcode GOTO Co1,Co2
1446 Co1:   Forlength=4-NOT Placesafter+Placesbefore+Placesafter
1448        Roomneeded=MAX(Forlength,LEN(Title$)+2)
1450        GOSUB Testforroom
1452        Print$=Print$&"N"&VAL$(Nonums)&","
1454        GOSUB Padleft
1456        Image$=Image$&"XM"
1458        IF NOT Placesbefore THEN 1462
1460        Image$=Image$&VAL$(Placesbefore)&"D"
1462        IF NOT Placesafter THEN 1466
1464        Image$=Image$&"."&VAL$(Placesafter)&"D"
1466        Image$=Image$&"X,"
1468        GOSUB Padright
1470        GOTO Nexti
1472 Co2:   Forlength=9-NOT Placesafter+Placesafter
```

```
1474        Roomneeded=MAX(Forlength,LEN(Title$)+2)
1476        GOSUB Testforroom
1478        Print$=Print$&"N"&VAL$(Nonums)&","
1480        GOSUB Padleft
1482        IF Placesafter THEN 1488
1484        Image$=Image$&"XMDEX"
1486        GOTO 1490
1488        Image$=Image$&"XMD."&VAL$(Placesafter)&"DEX,"
1490        GOSUB Padright
1492        GOTO Nexti
1494 S2:    IF L>Colperline-2 THEN Nexti
1496        Forlength=L+2
1498        Roomneeded=MAX(LEN(Title$),L)+2
1500        GOSUB Testforroom
1502        Print$=Print$&"S"&VAL$(Nostrs)&"$,"
1504        GOSUB Padleft
1506        Image$=Image$&"X"&VAL$(L)&"AX,"
1508        GOSUB Padright
1510        GOTO Nexti
1512 No2:   GOTO Nexti
1514 So2:   GOTO Nexti
1516 Nexti:NEXT I
1518        GOTO Nextwork
1520 Setb:  B$=VAL$(Linenumber)&"     "
1522        Linenumber=Linenumber+2
1524        RETURN
1526 Testforroom:  IF Roomneeded<=Roomleft THEN Roomenough
1528                GOSUB Flush
1530                GOTO Roomenough
1532 Flush:Image$[LEN(Image$)]=""     ! Get rid of the trailing commas
1534       Print$[LEN(Print$)]=""
1536       Imaget$[LEN(Imaget$)]=""
1538       Printt$[LEN(Printt$)]=""
15         GOSUB Setb
15         B$[6]=Print$
15         PRINT #5;B$
1546       GOSUB Setb
1548       B$[6]=Image$
1550       PRINT #5;B$
1552       B$=VAL$(Tlinestart)&" "&Printt$
1554       PRINT #5;B$
1556       Tlinestart=Tlinestart+1
1558       B$=VAL$(Tlinestart)&" "&Imaget$
1560       PRINT #5;B$
1562       Tlinestart=Tlinestart+1
1564       Imagecounter=Imagecounter+1
15         Titlecounter=Titlecounter+1
```

```
1568        Print$="PRINT USING Image"&VAL$(Imagecounter)&";"
1570        Printt$="PRINT USING Imaget"&VAL$(Titlecounter)&";"
1572        Image$="Image"&VAL$(Imagecounter)&": IMAGE "
1574        Imaget$="Imaget"&VAL$(Titlecounter)&": IMAGE "
1576        Usedflag=0
1578        Roomleft=Colperline
1580        RETURN
1582 Roomenough: CALL Leftjust(X,Before,After,Title$,Roomneeded)
1584                                    ! Center the title in its field
1586        Roomleft=Roomleft-Roomneeded  ! Decrement the amount of roo
m
1588                                      !  remaining in the line
1590        Printt$=Printt$&CHR$(34)&Title$&CHR$(34)&","
1592        Imaget$=Imaget$&VAL$(Before)&"X,"&VAL$(LEN(Title$))&"A,"&VA
L$(After)&"X,"
1594        Usedflag=1
1596        RETURN


 *** NEXT CHECK FOR STRINGS THAT WON'T FIT ON ONE LINE
 ***


1600 Nextwork:  IF Usedflag THEN GOSUB Flush
1602    FOR I=1 TO Howmanyfields
1604    Nonums=Nostrs=Nonummats=Nostrmats=0
1606    READ #1,1
1608    READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
1610    READ #3,A
1612    GOSUB Readj
1614    ON X GOTO N3,S3,Na3,Sa3
1616 N3:   GOTO Nexti1
1618 Na3:  GOTO Nexti1
1620 Sa3:  GOTO Nexti1
1622 S3:   IF L<=Colperline-2 THEN Nexti1
1624 !  ****** FIRST HANDLE THE TITLES OF THE LONG STRING ******
1626        GOSUB Titlehandle
1628        GOTO 1654
1630 Titlehandle:  B$=VAL$(Tlinestart)&" "&Printt$&CHR$(34)&Title$&CHR
$(34)
1632        PRINT #5;B$
1634        Tlinestart=Tlinestart+1
1636        CALL Leftjust(2,Before,After,Title$,Colperline)
1638        B$=VAL$(Tlinestart)&" "&Imaget$&VAL$(1)&"X,"&VAL$(LEN(Title
$))&"A,"&VAL$(After)&"X"
1640        PRINT #5;B$
1642        Tlinestart=Tlinestart+1
```

```
1644        Titlecounter=Titlecounter+1
1646        Printt$="PRINT USING Imaget"&VAL$(Titlecounter)&";"
1648        Imaget$="Imaget"&VAL$(Titlecounter)&": IMAGE "
1650        RETURN
1652 !  ****** NOW HANDLE THE ACTUAL PRINTING OF THE LONG STRING  ****
**
1654        Startsub=1
1656        Endsub=Colperline
1658        Print$=Print$&"S"&VAL$(Nostrs)&"$["&VAL$(Startsub)&","&VAL$
(Endsub)&"]"
1660        Image$=Image$&VAL$(Colperline)&"A"
1662        GOSUB Setb
1664        B$[6]=Print$
1666        PRINT #5;B$
1668        GOSUB Setb
1670        B$[6]=Image$
1672        PRINT #5;B$
1674        Imagecounter=Imagecounter+1
1676        Print$="PRINT USING Image"&VAL$(Imagecounter)&";"
1678        Image$="Image"&VAL$(Imagecounter)&": IMAGE "
1680        Roomneeded=Roomneeded-Colperline
1682        IF Roomneeded<=0 THEN Nexti1
1684        Startsub=Endsub+1
1686        Endsub=MIN(Endsub+Roomneeded,Endsub+Colperline)
1688        GOTO 1658
1690 Nexti1:  NEXT I


 ***  TAKE CARE OF ANY ARRAYS  ***


1694     Nonums=Nostrs=Nonummats=Nostrmats=0
1696     FOR I=1 TO Howmanyfields
1698     READ #1,1
1700     READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
1702     READ #3,A
1704     GOSUB Readj
1706     ON X GOTO N4,S4,Na4,Sa4
1708 N4:    GOTO Nexti2
1710 S4:    GOTO Nexti2
1712 Na4:   Varnam$="Na"&VAL$(Nonummats)
1714        GOTO Array
1716 Sa4:   Varnam$="Sa"&VAL$(Nostrmats)&"$"
1718 Array: ! ******  FIRST HANDLE THE TITLE(S) OF THE ARRAY(S) ******
1720        GOSUB Titlehandle
1722 ! ****** NOW HANDLE THE ACTUAL PRINTING OF THE ARRAYS ******
1724        GOSUB Setb
```

```
1726        B$[6]="PRINT "&Varnam$&"(*)"
1728        PRINT #5;B$
1730 Nexti2: NEXT I
1732 ! ****** FINISH TITLE SUBROUTINE ******
1734     PRINT #5;"200  "&Printt$[1,LEN(Printt$)-1]&"! Subroutine for p
rinting titles"
1736     PRINT #5;VAL$(Tlinestart)&" "&Printt$[1,LEN(Printt$)-1]
1738     Tlinestart=Tlinestart+1
1740     PRINT #5;VAL$(Tlinestart)&" "&Imaget$&VAL$(Colperline)&"("&CHR
$(34)&"-"&CHR$(34)&")"
1742     Tlinestart=Tlinestart+1
1744     PRINT #5;VAL$(Tlinestart)&" RETURN"
1746 ! ******  FINISH THE MAIN PROGRAM  ******
1748     GOSUB Setb
1750     B$[6]="PRINT LIN(2)"
1752     PRINT #5;B$
1754     GOSUB Setb
1756     B$[6]="GOTO Readnextpoint"
1758     PRINT #5;B$
1760     GOSUB Setb
1762     B$[6]="Done: PRINT LIN(2)"
1764     PRINT #5;B$
1766     GOSUB Setb
1768     B$[6]="PRINTER IS 16"
1770     PRINT #5;B$
1772     GOSUB Setb
1774     B$[6]="SUBEXIT"
1776     PRINT #5;B$
1778     SUBEXIT
1780 Padleft: P=Roomneeded-Forlength
1782           IF (P=0) OR (P=1) THEN RETURN
1784           P=INT(P/2)
1786           Image$=Image$&VAL$(P)&"X,"
1788           RETURN
1790 Padright:P=Roomneeded-Forlength
1792           IF P=0 THEN RETURN
1794           P=INT((P+1)/2)
1796           Image$=Image$&VAL$(P)&"X,"
1798           RETURN
1800 SUB Leftjust(W,B,A,T$,R)
1801 IF W-1 THEN 1807
1802 B=(R-LEN(T$))/2
1803 A=INT(B)+NOT (NOT FRACT(B))
1805 B=INT(B)
1806 SUBEXIT
1807 A=R-LEN(T$)-1
1808 B=1
1809 SUBEXIT
1810 SUB Sortwrite(#1,Sorttype,Sortlength,Keyflag)
1812 ! #1 is the data file
1814 DIM Sorttitle$[30],Title$[30],Dimstat$[80],Sortstat$[80],Readstat
```

```
$[160],Dim$[160],B$[160]
1815 READ #1,1
1816 READ #1;G,A,B,C,D,E,F
1818       READ #1,A       ! Set the pointer at the beginning of the tit
les
1820       PRINT PAGE,"Here are the fields in your data record:"
1822       READ #1;Title$
1824       PRINT SPA(5);Title$
1826       IF TYP(-1)<>3 THEN 1822
1828       READ #1,A
1830       INPUT "Please enter the title of the field you wish to sort
on",Sorttitle$
1832       Pos_sort=1
1834       READ #1;Title$
1836       IF Sorttitle$=Title$ THEN 1852
1838       IF TYP(-1)=3 THEN 1844
1840       Pos_sort=Pos_sort+1
1842       GOTO 1834
1844       BEEP
1846       DISP "TITLE NOT FOUND.  TRY AGAIN."
1848       WAIT 1000
1850       GOTO 1828
1852       Count=1


********************


1856       READ #1,1
1858       READ #1;G,A,B,C,D,E,F,K1,K2,K3,K4
1860 Readfor: READ #1;X
1862       IF Count=Pos_sort THEN GOTO Sortassignment
1864       Count=Count+1
1866       ON X GOSUB Nu,St,Na,Sa
1868       GOTO Readfor


********************


1872 REM        X,Y,Z,S,T, and I are all destroyed here
1874 Nu:        RETURN
1876 St:        READ #1;X
1878            RETURN
1880 Na:        READ #1;X
1882            FOR I=1 TO X
1884            READ #1;S,T
```

```
1886            NEXT I
1888            RETURN
1890 Sa:        READ #1;X,Y
1892            FOR I=1 TO Y
1894            READ #1;S,T
1896            NEXT I
1898            RETURN


********************


1902 Sortassignment: Varindex=0
1904            READ #1,1
1906            READ #1;G,A,B,C,D,E,F,K1,K2,K3,K4
1908            FOR I=1 TO Count-1
1910                 READ #1;Y
1912                 IF Y=X THEN Varindex=Varindex+1
1914                 ON Y GOSUB Read1,Read2,Read3,Read4
1916            NEXT I
1918            READ #1;Y
1920            ON X GOTO Key1,Key2,Key3,Key4


********************


1924 Key1:           Sortstat$="Sort=N"&VAL$(Varindex+1)&"    !  "&Sortt
itle$
1926                 Sorttype=1
1928                 Recordlength=20
1930                 Sortlength=1
1932                 Dimstat$="DIM Sort$[1]"
1934                 GOTO Write
1936 Key2:           READ #1;Z
1938                 Sorttype=2
1940                 Dimstat$="DIM Sort$["&VAL$(Z)&"]"
1942                 Sortstat$="Sort$=S"&VAL$(Varindex+1)&"$  !  "&Sort
title$
1944                 Recordlength=2*INT((Z+1)/2)+14
1946                 Sortlength=Z
1948                 GOTO Write
1950 Key3:           READ #1;Z
1952                 Sorttype=1
1954                 Sortstat$="Sort=Na"&VAL$(Varindex+1)&"("
1956                 Dimstat$="DIM Sort$[1]"
1958                 Recordlength=20
```

```
1960                Sortlength=1
1962                GOTO Loopsort
1964 Key4:          READ #1;Z
1966                Sorttype=2
1968                Recordlength=2*INT((Z+1)/2)+14
1970                Sortlength=Z
1972                Dimstat$="DIM Sort$["&VAL$(Z)&"]"
1974                Sortstat$="Sort$=So"&VAL$(Varindex+1)&"$("
1976                READ #1;Z
1978 Loopsort:      FOR J=1 TO Z
1980                     READ #1;S,T
1982                     DISP "Please enter subscript #";J;" for the s
ort field";
1984                     INPUT "",Y
1986                     IF (S<=Y) AND (Y<=T) THEN Okay2
1988                     BEEP
1990                     DISP "Subscript range is from";S;"to";T;"----
-";
1992                     GOTO 1982
1994 Okay2:              Sortstat$=Sortstat$&VAL$(Y)&","
1996                NEXT J
1998                Sortstat$[LEN(Sortstat$)]=")    !  "&Sorttitle$
2000                GOTO Write

********************

2004 Read1:         RETURN
2006 Read2:         READ #1;Z
2008                RETURN
2010 Read3:         READ #1;Z
2012                GOTO 2016
2014 Read4:         READ #1;Y,Z
2016                FOR K=1 TO Z
2018                     READ #1;S,T
2020                NEXT K
2022                RETURN

********************

2026 Write:      ! At this point, Dimstat$ holds the DIM statement for
the sort
2028             !  field, and Sortstat$ holds the assignment of the so
rt field.
```

```
2030             !  Sorttype=1 is for a numeric sort, Sorttype=2 is for
 a string.
2032             !  Recordlength tells how long each record of the sort
 file will
2034             !  have to be in order to hold the sort field value an
d a its
2036             !  corresponding record pointer.
2038 ! WRITE A MERGE ROUTINE
2040 ASSIGN #5 TO "DBTMP6",Check
2042 IF Check THEN 2046
2044 PURGE "DBTMP6"
2046 CREATE "DBTMP6",20
2048 ASSIGN #5 TO "DBTMP6"
2050 BUFFER #5
2052 PRINT #5;"10   SUB Merge(#9,#2,#5,Maxsize,Dirnorecs,Keylength,Key
sinmemory)"
2054 PRINT #5;"15   ! This subprogram merges the directory records to
onefile for"
2056 PRINT #5;"20   !  report generation."
2058 PRINT #5;"22   !  #9 - the data file"
2060 PRINT #5;"25   !  #2 - the directory"
2062 PRINT #5;"30   !  #5 - the master file"
2064 PRINT #5;"35   !  Maxsize - the number of records in the master f
ile"
2066 PRINT #5;"40   !  Dirnorecs - the number of records in the direct
ory"
2068 PRINT #5;"50   !  Keylength - the length of the key field"
2070 PRINT #5;"55   !  Keysinmemory - the number of keys allowed in me
mory"
2072 PRINT #5;"65   OPTION BASE 1"
2074 IF Keyflag THEN 2080
2076 PRINT #5;"70   DIM Keys$(Keysinmemory)[Keylength],Pointers(Keysin
memory)"
2078 GOTO 2082
2080 PRINT #5;"70   DIM Keys(Keysinmemory),Pointers(Keysinmemory)"
2082 PRINT #5;"71   "&Dimstat$
2084 READ #1,D      ! Set the serial pointer to read the DIM statement
s
2086 Statno=72
2088 READ #1;Dim$
2090 GOSUB Setb
2092 B$[6]=Dim$
2094 PRINT #5;B$
2096 IF TYP(-1)<>3 THEN 2088
2098 PRINT #5;"88   Count=1"
2100 PRINT #5;"89   Reclen="&VAL$(Recordlength)
2102 PRINT #5;"90   ASSIGN #5 TO "&CHR$(34)&"DBTMP7"&CHR$(34)&",C"
```

```
2104 PRINT #5;"95   IF NOT (C-1) THEN 105"
2106 PRINT #5;"100  PURGE "&CHR$(34)&"DBTMP7"&CHR$(34)
2108 PRINT #5;"105  CREATE "&CHR$(34)&"DBTMP7"&CHR$(34)&",Maxsize,Recl
en"
2110 PRINT #5;"110  ASSIGN #5 TO "&CHR$(34)&"DBTMP7"&CHR$(34)
2112 PRINT #5;"115  BUFFER #5"
2114 PRINT #5;"120  Totalkeys=0   ! Initialize the key counter"
2116 PRINT #5;"125  FOR I=1 TO Dirnorecs"
2118 IF Keyflag THEN 2124
2120 PRINT #5;"130     READ #2,I;Keysinmemory,Keysused,Overflow,Keys*(
*),Pointers(*)"
2122 GOTO 2126
2124 PRINT #5;"130     READ #2,I;Keysinmemory,Keysused,Overflow,Keys(*
),Pointers(*)"
2126 PRINT #5;"135     FOR J=1 TO Keysused"
2128 PRINT #5;"140     READ #9,Pointers(J)"
2130 READ #1,C    ! Set the serial pointer to read the READ # statemen
ts
2132 Statno=141
2134 READ #1;Readstat$
2136 GOSUB Setb
2138 B$[6]=Readstat$
2140 PRINT #5;B$
2142 IF TYP(-1)<>3 THEN 2134
2144 PRINT #5;"155        "&Sortstat$
2146 ON Sorttype GOTO 2148,2152
2148 PRINT #5;"160        PRINT #5,Count;Sort,Pointers(J)"
2150 GOTO 2154
2152 PRINT #5;"160        PRINT #5,Count;Sort$,Pointers(J)"
2154 PRINT #5;"165        Count=Count+1"
2156 PRINT #5;"170     NEXT J"
2158 PRINT #5;"175     Totalkeys=Totalkeys+Keysused"
2160 PRINT #5;"180  NEXT I"
2162 ON Sorttype GOTO 2164,2168
2164 PRINT #5;"182  CALL Vectorexsort_q(1,Totalkeys,#5)"
2166 GOTO 2170
2168 PRINT #5;"182  CALL Stringexsort_q(1,Totalkeys,"&VAL$(Sortlength)
&",#5)"
2170 PRINT #5;"185  SUBEXIT"
2172 SUBEXIT
2174 Setb: B$=VAL$(Statno)&"    "
2176       Statno=Statno+1
2178       RETURN
2180 SUBEND
```

## Listing of File "DBLTKY"

```
10    SUB Listkeys(#2,Dirnorecs,Keysinmemory,Keyflag,Keylength)
15    IF Keyflag THEN N
20  S: CALL Ls(#2,Dirnorecs,Keysinmemory,Keyflag,Keylength)
25     SUBEXIT
30  N: CALL Ln(#2,Dirnorecs,Keysinmemory,Keyflag)
35     SUBEND
40    SUB Ls(#2,Dirnorecs,Keysinmemory,Keyflag,Keylength)
45    OPTION BASE 1
50    DIM Keys$(Keysinmemory)[Keylength],Pointers(Keysinmemory)
55    FOR I=1 TO Dirnorecs
60       READ #2,I;Keysinmemory,Keysused,Overflow,Keys$(*),Pointers(*)
65       FOR J=1 TO Keysused
70          PRINT Keys$(J)
75       NEXT J
80    NEXT I
85    SUBEXIT
90    SUB Ln(#2,Dirnorecs,Keysinmemory,Keyflag)
95    OPTION BASE 1
100   DIM Keys(Keysinmemory),Pointers(Keysinmemory)
105   FOR I=1 TO Dirnorecs
110      READ #2,I;Keysinmemory,Keysused,Overflow,Keys(*),Pointers(*)
115      FOR J=1 TO Keysused
120         PRINT Keys(J)
125      NEXT J
130   NEXT I
135   SUBEXIT
```

## Listing of File "DBSUB1"

```
2350 SUB Findemptyrecord(#2,Recno)
2355 ! This subprogram returns the defined record number of a free
2360 !  record of the data base file in the variable Recno.
2365 READ #2,1;Freepoint
2370 READ #2,Freepoint;Recno,X
2375 IF Recno=0 THEN SUBEXIT
2380 READ #2,Recno;Nextpoint
2385 PRINT #2,Freepoint;Nextpoint,X
2390 SUBEND



2400 SUB Returnrecord(#2,Recno)
2405 ! This subprogram returns the defined record number (given by
2410 !  the variable Recno) to the available list.
2415 READ #2,1;Freepoint
2420 READ #2,Freepoint;Nextpoint,X
2425 PRINT #2,Freepoint;Recno,X
2430 PRINT #2,Recno;Nextpoint
2435 SUBEND



2440 SUB Updatepage(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,Keyspe
rrecord,Dirnorecs,Collide)
2445 IF Keyflag THEN Numeric
2450 String: CALL Upstring(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag
,Keysperrecord,Dirnorecs,Collide)
2455          SUBEXIT
2460 Numeric:CALL Upnum(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,Ke
ysperrecord,Dirnorecs,Collide)
2465          SUBEND
2470 SUB Upstring(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,Keysperr
ecord,Dirnorecs,Collide)
2475 OPTION BASE 1
2480 DIM Key$(Keysperrecord)[Keylength],Key(1),Pointers(Keysperrecord)
2485 CALL Update(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,Keysperre
cord,Key$(*),Key(*),Pointers(*),Dirnorecs,Collide)
2490 SUBEND
2495 SUB Upnum(#2,Recno,Key,Key$,Keylengtn,Pointer,Keyflag,Keysperreco
rd,Dirnorecs,Collide)
2500 OPTION BASE 1
2505 DIM Key(Keysperrecord),Key$(1)[1],Pointers(Keysperrecord)
2510 CALL Update(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,Keysperre
cord,Key$(*),Key(*),Pointers(*),Dirnorecs,Collide)
```

```
2515 SUBEND



2520 SUB Update(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,Keysperrec
ord,Keys$(*),Keys(*),Pointers(*),Dirnorecs,Collide)
2525 ! This subprogram inserts a key and its record number into the
2530 !  proper record in the directory.
2535 !  #2 -> The directory file
2540 !  Recno -> The page number of the proper record
2545 !  Key, Key$, Pointer -> The key to be inserted and its pointer
2550 !  Keyflag=0 -> String key.  Keyflag=1 -> Numeric key.
2555 !  Maxsize -> The number of keys and pointers in memory.
2560 !  Keys$(*), Keys(*) -> Key arrays
2565 !  Dirnorecs -> The number of pages in the directory.
2570 Test:     IF Keyflag THEN Numerics
2575 Strings: READ #2,Recno;Keysperrecord,Keysused,Overflow,Keys$(*),P
ointers(*)
2580           CALL Binsearch_s(Keys$(*),Key$,Keysused,Found,Where)
2585           IF NOT Found THEN 2600
2590           Collide=1
2595           SUBEXIT
2600           IF Keysused<Keysperrecord THEN Found
2605           PRINT #2,Recno;Keysperrecord,Keysused,1,Keys$(*),Pointer
s(*)
2610           GOTO Outofroom
2615 Found:    FOR I=Keysused TO Where+1 STEP -1
2620           Keys$(I+1)=Keys$(I)
2625           Pointers(I+1)=Pointers(I)
2630           NEXT I
2635           Keys$(Where+1)=Key$
2640           Pointers(Where+1)=Pointer
2645           PRINT #2,Recno;Keysperrecord,Keysused+1,Overflow,Keys$(*
),Pointers(*)
2650           SUBEXIT
2655 Numerics:READ #2,Recno;Keysperrecord,Keysused,Overflow,Keys(*),Po
inters(*)
2660           CALL Binsearch_n(Keys(*),Key,Keysused,Found,Where)
2665           IF NOT Found THEN 2680
2670           Collide=1
2675           SUBEXIT
2680           IF Keysused<Keysperrecord THEN Found1
2685           PRINT #2,Recno;Keysperrecord,Keysused,1,Keys(*),Pointers
(*)
2690           GOTO Outofroom
2695 Found1:   FOR I=Keysused+1 TO Where+1 STEP -1
```

```
2700            Keys(I+1)=Keys(I)
2705            Pointers(I+1)=Pointers(I)
2710            NEXT I
2715            Keys(Where+1)=Key
2720            Pointers(Where+1)=Pointer
2725            PRINT #2,Recno;Keysperrecord,Keysused+1,Overflow,Keys(*)
,Pointers(*)
2730            SUBEXIT
2735 Outofroom:Recno=Recno+1
2740            IF Recno>Dirnorecs THEN Recno=1
2745            CALL Update(#2,Recno,Key,Key$,Keylength,Pointer,Keyflag,
Keysperrecord,Keys$(*),Keys(*),Pointers(*),Dirnorecs,Collide)
2750            SUBEXIT



2755 SUB Garbagecollect(Keys(*),Keys$(*),Pointers(*),Keyflag,Keysused,
Keysperrecord,Deletesubscript)
2760 ! This subprogram will delete the key and pointer at the
2765 !  position given by Deletesubscript by bumping everything
2770 !  above it down one.
2775 !  Keysused gets decremented by one at the end of the routine
2780 !  left at the end of the array.
2785 !  Keyflag=1 -> Numeric keys.   Keyflag=0 -> Sting keys.
2790 D=Deletesubscript
2795                IF NOT Keyflag THEN Strings
2800 Numerics:      FOR I=D TO Keysused-1
2805                Keys(I)=Keys(I+1)
2810                Pointers(I)=Pointers(I+1)
2815                NEXT I
2820                Keys(Keysused)=9E99
2825                Pointers(Keysused)=0
2830                GOTO Dec
2835 Strings:       FOR I=D TO Keysused-1
2840                Keys$(I)=Keys$(I+1)
2845                Pointers(I)=Pointers(I+1)
2850                NEXT I
2855                Keys$(Keysused)=CHR$(127)
2860                Pointers(Keysused)=0
2865 Dec:           Keysused=Keysused-1
2870                SUBEND



2875 SUB Hashpage(Pageno,Key,Key$,Norecs,Keyflag)
```

```
2880 ! This subprogram hashes the key to find the page number
2885 !  where the key belongs.
2890 !  Keyflag=1 ==> the key is numeric.
2895 !  Keyflag=0 ==> the key is a string.
2900 !  Norecs=the number of records in the directory.
2905 Hold=Key
2910 IF Keyflag THEN Number
2915 String: Hold=0
2920         Keylength=LEN(Key$)
2925         Step=Keylength/6
2930         Step=INT(Step)+(FRACT(Step)<>0)
2935         FOR I=Keylength TO 1 STEP -Step
2940         Hold=Hold+NUM(Key$[I])*10^(Keylength-I)
2945         NEXT I
2950 Number: IF Hold=0 THEN Hold=1E-6
2955         H=ABS(LOG(ABS(Hold)))
2960         IF H=0 THEN H=7
2965         RANDOMIZE -1/H
2970         Pageno=INT(RND*Norecs)+1
2975         SUBEND
```

```
2980 SUB Getkey(#1,Key,Key$,Keyflag,Delete_or_edit$)
2985 ! This subprogram asks the user for the key of the record
2990 !  he wants to delete or edit.
2995 !  Keyflag=0 -> string.  Keyflag=1 -> numeric
3000 IF Keyflag THEN N
3005 S:Type$="string"
3010  GOTO Prompt
3015 N:Type$="numeric"
3020 Prompt:PRINT PAGE;"  Please enter the key of the record you wish
to "&Delete_or_edit$&"."
3025       READ #1,1
3030       READ #1;G,A,B,C,D,E,F,Keytype,Keylength,Maxsize,Poskey
3035       READ #1,A
3040       FOR I=1 TO Poskey
3045       READ #1;T$
3050       NEXT I
3055       PRINT "The title of the key field is "&T$&" and it is a "
3060       PRINT Type$&" field."
3065 IF Keyflag THEN N1
3070 S1: LINPUT "Please enter the string value for the key",Key$
3071     PRINT LIN(2),"KEY VALUE:  "&Key$
3072     A$=""
3073     INPUT "Did you enter the correct key value?",A$
```

```
3074      IF UPC$(A$)="N" THEN 3078
3075      IF UPC$(A$)="Y" THEN 3080
3076      BEEP
3077      GOTO 3072
3078      EDIT "Press CONT when you have made your corrections",Key$
3079      GOTO 3071
3080      DISP "Thank you"
3085      WAIT 500
3090      SUBEXIT
3095 N1:  INPUT "Please enter the numeric value for the key",Key
3100      PRINT LIN(2),"KEY VALUE:  "&VAL$(Key)
3115      A$=""
3116      INPUT "Did you enter the correct key value?",A$
3117      IF UPC$(A$)="N" THEN N1
3118      IF UPC$(A$)="Y" THEN 3125
3119      BEEP
3120      GOTO 3115
3125      DISP "Thank you"
3130      WAIT 500
3135      SUBEXIT
```

```
3140 SUB Findkey(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysinmemory,
Dirnorecs,Maxsize,Switch)
3145 IF Keyflag THEN N
3150 S:CALL Fs(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysinmemory,Di
rnorecs,Maxsize,Switch)
3155 SUBEXIT
3160 N:CALL Fn(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysinmemory,Di
rnorecs,Maxsize,Switch)
3165 SUBEND
3170 SUB Fs(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysinmemory,Dirno
recs,Maxsize,Switch)
3175 OPTION BASE 1
3180 DIM Keys$(Keysinmemory)[Keylength],Pointers(Keysinmemory),Keys(1)
3185 CALL F(#1,#2,Pageno,Key,Key$,Keylength,Keys$(*),Keys(*),Pointers(
*),Keyflag,Dirnorecs,Maxsize,Switch)
3190 SUBEXIT
3195 SUB Fn(#1,#2,Pageno,Key,Key$,Keylength,Keyflag,Keysinmemory,Dirno
recs,Maxsize,Switch)
3200 OPTION BASE 1
3205 DIM Keys$(1)[1],Pointers(Keysinmemory),Keys(Keysinmemory)
3210 CALL F(#1,#2,Pageno,Key,Key$,Keylength,Keys$(*),Keys(*),Pointers(
*),Keyflag,Dirnorecs,Maxsize,Switch)
3215 SUBEXIT
```

```
3220 SUB F(#1,#2,Pageno,Key,Key$,Keylength,Keys$(*),Keys(*),Pointers(*
),Keyflag,Dirnorecs,Maxsize,Switch)
3225 ! #1 is the data file
3230 ! #2 is the directory file
3235 ! Pageno is the record in the directory that the key hashed to
3240 ! Key or Key$ has the key
3245 ! Keylength is the length of the key
3250 ! Keys(*) or Keys$(*) will have the page of the directory
3255 ! Pointers(*) keeps the records associated with the keys
3260 ! Keyflag=1 -> numeric key.  Keyflag=0 -> string key.
3265 ! Dirnorecs = number of records in the directory file
3270 ! Maxsize = Total capacity of the data file
3275 ! Switch=0 -> Delete;     Switch=1 -> Insert
3280 DIM Ktem$[Keylength]
3285 IF Keyflag THEN N
3290 S: READ #2,Pageno;Keysinmemory,Keysused,Overflow,Keys$(*),Pointer
s(*)
3295    CALL Binsearch_s(Keys$(*),Key$,Keysused,Found,Where)
3300    IF Found=0 THEN Notfound
3305    Ktem$=Key$
3310    Recno=Pointers(Where)
3315    IF NOT Switch THEN Change
3320    CALL Edit(#1,Recno,Key,Key$,Keyflag)
3325    IF Ktem$<>Key$ THEN Change   ! See if the key was changed
3330    SUBEXIT
3335 Notfound: IF NOT Overflow THEN Un
3340    Pageno=Pageno+1
3345    IF Pageno>Dirnorecs THEN Pageno=1
3350    IF Keyflag THEN N
3355    GOTO S
3360 Un:IF Keyflag THEN 3375
3365    PRINT LIN(2);"There is no record having the key "&Key$&"."
3370    SUBEXIT
3375    PRINT LIN(2);"There is no record having the key "&VAL$(Key)&".
"
3380    SUBEXIT
3385 N: READ #2,Pageno;Keysinmemory,Keysused,Overflow,Keys(*),Pointers
(*)
3390    CALL Binsearch_n(Keys(*),Key,Keysused,Found,Where)
3395    IF Found=0 THEN Notfound
3400    Ktem=Key
3405    Recno=Pointers(Where)
3410    IF NOT Switch THEN Change
3415    CALL Edit(#1,Recno,Key,Key$,Keyflag)
3420    IF Ktem<>Key THEN Change   ! See if the key was changed.
3425    SUBEXIT
3430 Change:CALL Garbagecollect(Keys(*),Keys$(*),Pointers(*),Keyflag,K
```

```
eysused,Keysinmemory,Where)
3435     IF Keyflag THEN 3450
3440     PRINT #2,Pageno;Keysinmemory,Keysused,Overflow,Keys$(*),Pointe
rs(*)
3445     GOTO 3455
3450     PRINT #2,Pageno;Keysinmemory,Keysused,Overflow,Keys(*),Pointer
s(*)
3455     IF NOT Switch THEN Delete
3460     CALL Hashpage(Pageno,Key,Key$,Dirnorecs,Keyflag)
3465     CALL Updatepage(#2,Pageno,Key,Key$,Keylength,Recno,Keyflag,Key
sinmemory,Dirnorecs,Collide)
3470     IF NOT Collide THEN 3490
3475     Collide=0
3480     CALL Replacekey(#1,Recno,Key,Key$,Keyflag)
3485     GOTO 3460
3490     SUBEXIT
3495 Delete:CALL Returnrecord(#1,Recno)
3500     IF Keyflag THEN Nmes
3505 Smes:  PRINT LIN(2),"The record having the key "&Key$&" has been
deleted."
3510     SUBEND
3515 Nmes:  PRINT LIN(2),"The record having the key "&VAL$(Key)&" has
been deleted."
3520     SUBEND



3525 SUB Binsearch_s(Keys$(*),Key$,Keysused,Found,Where)
3530 ! Algorith C, p. 412, Knuth, Sorting and Searching
3535 ! If Key$ is found, then Found=1 and Where=where
3540 ! If Key$ is not found, then Found=0 and Where=the position immed
iately
3545 !  below where the key belongs
3550 IF NOT Keysused THEN 3570
3555 Dim=INT(LOG(Keysused)/LOG(2))+2
3560 CALL Bs(Keys$(*),Key$,Keysused,Found,Where,Dim)
3565 SUBEXIT
3570 Found=0
3575 Where=0
3580 SUBEND
3585 SUB Bs(Keys$(*),Key$,Keysused,Found,Where,Dim)
3590 OPTION BASE 1
3595 DIM Delta(Dim)
3600 FOR J=1 TO Dim
3605 Delta(J)=INT(Keysused/2^J+.5)
3610 NEXT J
```

```
3615 C1:   I=Delta(1)
3620       J=2
3625 C2:   IF NOT I THEN Notfound
3630       IF Key$<Keys$(I) THEN C3
3635       IF Key$>Keys$(I) THEN C4
3640       Where=I   ! Here the two are equal, and I is the subscript
3645       Found=1
3650       SUBEXIT
3655 C3:   IF Delta(J)=0 THEN Notfound3
3660       I=I-Delta(J)
3665       J=J+1
3670       GOTO C2
3675 C4:   IF Delta(J)=0 THEN Notfound4
3680       I=I+Delta(J)
3685       J=J+1
3690       GOTO C2
3695 Notfound3:Where=I-1
3700       GOTO Notfound
3705 Notfound4:Where=I
3710 Notfound:Found=0
3715            SUBEND



3720 SUB Binsearch_n(Keys(*),Key,Keysused,Found,Where)
3725 ! Algorith C, p. 412, Knuth, Sorting and Searching
3730 ! If Key is found, then Found=1 and Where=where
3735 ! If Key is not found, then Found=0 and Where=the position immedi
ately
3740 !  below where the key belongs
3745 IF NOT Keysused THEN 3765
3750 Dim=INT(LOG(Keysused)/LOG(2))+2
3755 CALL Bn(Keys(*),Key,Keysused,Found,Where,Dim)
3760 SUBEXIT
3765 Found=0
3770 Where=0
3775 SUBEND
3780 SUB Bn(Keys(*),Key,Keysused,Found,Where,Dim)
3785 OPTION BASE 1
3790 DIM Delta(Dim)
3795 FOR J=1 TO Dim
3800 Delta(J)=INT(Keysused/2^J+.5)
3805 NEXT J
3810 C1:   I=Delta(1)
3815       J=2
3820 C2:   IF NOT I THEN Notfound
```

```
3825       IF Key<Keys(I) THEN C3
3830       IF Key>Keys(I) THEN C4
3835       Where=I  ! Here the two must be equal, and I is the subscrip
t
3840       Found=1
3845       SUBEXIT
3850 C3:   IF Delta(J)=0 THEN Notfound3
3855       I=I-Delta(J)
3860       J=J+1
3865       GOTO C2
3870 C4:   IF Delta(J)=0 THEN Notfound4
3875       I=I+Delta(J)
3880       J=J+1
3885       GOTO C2
3890 Notfound3:Where=I-1
3895       GOTO Notfound
3900 Notfound4:Where=I
3905 Notfound:Found=0
3910           SUBEND
```

## Listing of File "DBEXSS"

```
10  SUB Stringexsort_q(I1,J1,Length,#5)
15   ! This subprogram sorts the file #5 in ascending order from
20   !  records I1 through J1 on the key.  Corresponding pointers get
25   !  carried along.
30        N=J1+1-I1
31        IF N<2 THEN 45
35        Logtwo=INT(LGT(N)/LGT(2))+1
40        CALL Qsort1(Logtwo,I1,J1,Length,#5)
45  SUBEXIT
50  SUB Qsort1(Log,I1,J1,Length,#5)
55   OPTION BASE 1
60   DIM L(Log),U(Log),T$[Length],T1$[Length],Ai$[Length],Aj$[Length],
Al$[Length],Ak$[Length]
65        M=1
70        I=I1
75        J=J1
80        READ #5,I;Ai$,Pi
85        READ #5,J;Aj$,Pj
90 Start1:  IF I>=J THEN Nextgroup
95 Start2:  K=I
100         Ak$=Ai$
105         Pk=Pi
110         I2=INT((J+I)/2)
115         READ #5,I2;T$,Pt
120 I1:     IF Ai$<=T$ THEN Lowmiddle1
125         PRINT #5,I2;Ai$,Pi
130         PRINT #5,I;T$,Pt
135         Ai$=T$
140         Pi=Pt
145         READ #5,I2;T$,Pt
150 Lowmiddle1:  L=J
155              Al$=Aj$
160              Pl=Pj
165 I2:          IF Aj$>=T$ THEN Middlehigh
170              PRINT #5,I2;Aj$,Pj
175              PRINT #5,J;T$,Pt
180              Aj$=T$
185              Pj=Pt
190              READ #5,I2;T$,Pt
195 I3:          IF Ai$<=T$ THEN Middlehigh
200              PRINT #5,I2;Ai$,Pi
205              PRINT #5,I;T$,Pt
210              Ai$=T$
215              Pi=Pt
220              READ #5,I2;T$,Pt
225 Middlehigh:  L=L-1
230              READ #5,L;Al$,Pl
```

```
235 I4:           IF A1$>T$ THEN Middlehigh
240               T1$=A1$
245               Pt1=P1
250 Stepup:  K=K+1
255          READ #5,K;Ak$,Pk
260 I5:      IF Ak$<T$ THEN Stepup
265          IF K>L THEN Passed
270          PRINT #5,L;Ak$,Pk
275          A1$=Ak$
280          P1=Pk
285          PRINT #5,K;T1$,Pt1
290          Ak$=T1$
295          Pk=Pt1
300          GOTO Middlehigh
305 Passed:  IF L-I<=J-K THEN Storehigh
310          L(M)=I
315          U(M)=L
320          I=K
325          Ai$=Ak$
330          Pi=Pk
335          M=M+1
340          GOTO 375
345 Storehigh:  L(M)=K
350             U(M)=J
355             J=L
360             Aj$=A1$
365             Pj=P1
370             M=M+1
375             IF J-I>=11 THEN Start2
380             IF I=I1 THEN Start1
385             I=I-1
390 Inc:   I=I+1
395        READ #5,I;Ai$,Pi
400        IF I=J THEN Nextgroup
405        READ #5,I+1;T$,Pt
410 I6:    IF Ai$<=T$ THEN Inc
415        K=I
420        Ak$=Ai$
425        Pk=Pi
430 Copy:   PRINT #5,K+1;Ak$,Pk
435         K=K-1
440         READ #5,K;Ak$,Pk
445 I7:     IF T$<Ak$ THEN Copy
450         PRINT #5,K+1;T$,Pt
455         GOTO Inc
460 Nextgroup:  M=M-1
465             IF M=0 THEN Out
```

```
470              I=L(M)
475              J=U(M)
480              READ #5,I;Ai$,Pi
485              READ #5,J;Aj$,Pj
490              GOTO 375
495 Out:  SUBEXIT
```

## Listing of File "DBEXSN"

```
10  SUB Vectorexsort_q(I1,J1,#5)
15   ! This subprogram sorts the file #5 in ascending order from,
20   !  records I1 through J1 on the key.  Corresponding pointers get
25   !  carried along.
30       N=J1+1-I1
31       IF N<2 THEN 45
35       Logtwo=INT(LGT(N)/LGT(2))+1
40       CALL Qsort2(Logtwo,I1,J1,#5)
45  SUBEXIT
50  SUB Qsort2(Log,I1,J1,#5)
55   OPTION BASE 1
60   DIM L(Log),U(Log)
65       M=1
70       I=I1
75       J=J1
80       READ #5,I;Ai,Pi
85       READ #5,J;Aj,Pj
90 Start1:  IF I>=J THEN Nextgroup
95 Start2:  K=I
100         Ak=Ai
105         Pk=Pi
110         I2=INT((J+I)/2)
115         READ #5,I2;T,Pt
120 I1:     IF Ai<=T THEN Lowmiddle1
125         PRINT #5,I2;Ai,Pi
130         PRINT #5,I;T,Pt
135         Ai=T
140         Pi=Pt
145         READ #5,I2;T,Pt
150 Lowmiddle1:  L=J
155              Al=Aj
160              Pl=Pj
165 I2:          IF Aj>=T THEN Middlehigh
170              PRINT #5,I2;Aj,Pj
175              PRINT #5,J;T,Pt
180              Aj=T
185              Pj=Pt
190              READ #5,I2;T,Pt
195 I3:          IF Ai<=T THEN Middlehigh
200              PRINT #5,I2;Ai,Pi
205              PRINT #5,I;T,Pt
210              Ai=T
215              Pi=Pt
220              READ #5,I2;T,Pt
225 Middlehigh:  L=L-1
230              READ #5,L;Al,Pl
235 I4:          IF Al>T THEN Middlehigh
```

```
240                T1=A1
245                Pt1=P1
250 Stepup:   K=K+1
255           READ #5,K;Ak,Pk
260 I5:       IF Ak<T THEN Stepup
265           IF K>L THEN Passed
270           PRINT #5,L;Ak,Pk
275           A1=Ak
280           P1=Pk
285           PRINT #5,K;T1,Pt1
290           Ak=T1
295           Pk=Pt1
300           GOTO Middlehigh
305 Passed:   IF L-I<=J-K THEN Storehigh
310           L(M)=I
315           U(M)=L
320           I=K
325           Ai=Ak
330           Pi=Pk
335           M=M+1
340           GOTO 375
345 Storehigh:  L(M)=K
350               U(M)=J
355               J=L
360               Aj=A1
365               Pj=P1
370               M=M+1
375               IF J-I>=11 THEN Start2
380               IF I=I1 THEN Start1
385               I=I-1
390 Inc:   I=I+1
395        READ #5,I;Ai,Pi
400        IF I=J THEN Nextgroup
405        READ #5,I+1;T,Pt
410 I6:    IF Ai<=T THEN Inc
415        K=1
420        Ak=Ai
425        Pk=Pi
430 Copy:   PRINT #5,K+1;Ak,Pk
435         K=K-1
440         READ #5,K;Ak,Pk
445 I7:     IF T<Ak THEN Copy
450         PRINT #5,K+1;T,Pt
455         GOTO Inc
460 Nextgroup:  M=M-1
465               IF M=0 THEN Out
470               I=L(M)
```

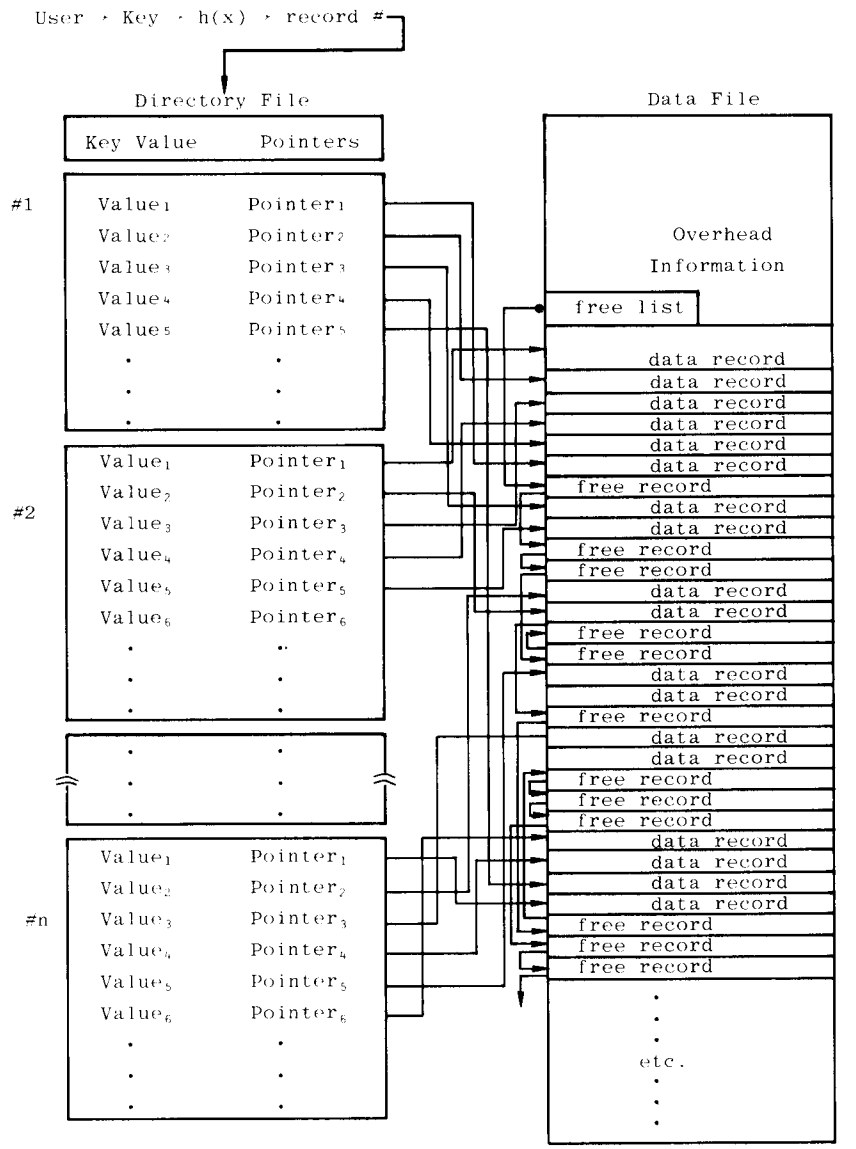
```
475                J=U(M)
480                READ #5,I;Ai,Pi
485                READ #5,J;Aj,Pj
490                GOTO 375
495 Out:   SUBEXIT
```

HEWLETT hp PACKARD

# EXPLANATION OF FILE STRUCTURES

This part of the manual explains how the Information Management programs structure data base. If the user wishes to modify the programs (or write his own programs) in order to deal with a specific problem, he should understand the material in this section thoroughly.

The following diagram gives a pictorial overview of the file structures used in the data base:



1. The user specifies the key value
2. The key value is fed to a "hashing function".
3. The value returned from the "hashing function" specifies a record # in the directory file.
4. The directory record is read in.
5. The pointer which corresponds to the key value in the directory record gives the location of the corresponding data record.

When the user initializes his data file (by running the program in file DBINIT),certain information is stored in the first few records of the file. This information is necessary to the programs that access the data file.

First of all, it is necessary to keep track of all the fields that the user specifies for each record in his data base. There are four types of fields possible: Simple numeric fields, simple string fields, numeric arrays, and string arrays. In order for the system to be able to read and print the data from and to mass storage devices, the initialization program builds PRINT # and READ # statements as the user specifies his fields. Also, DIM statements are built for the array fields that are needed. The initialization program assigns variables to each field in the data record for use in the PRINT #, READ #, and DIM statements. The rules governing the assignment of variable names follow:

For the ith simple numeric field, the variable name will be Ni, where i is some number.
For the ith simple string field, the variable name will be Si$, where i is some number.
For the ith numeric array, the variable name will be Nai(*), where i is some number.
For the ith string array, the variable name will be Sai$(*), where i is some number.

Example 1:

Suppose the user directs the initialization program to build a data record in the following manner:

1. Numeric field (Title: Employee number)
2. String field (Name)
3. String field (Address)
4. String field (City, State)
5. Numeric array (Payroll deductions) (3x3 array)
6. Numeric field (Zip code)
7. String array (Names of dependents) (1x7 array)
8. Numeric array (ages of dependents) (1x7 array)

For the given example, the initialization program would construct the following statements:

```
DIM  Na1(1:3,1:3),   Sal$(1:7), Na2(1:7)

READ #9; N1, S1, S2, S3, Na1(*), N2, Sal$(*), Na2(*)

PRINT #9; N1, S1, S2, S3, Na1(*), N2, Sal$(*), Na2(*)
```

The DIM, READ #, and PRINT # statements are all stored in the data file where they can be used later by other programs. Exactly where they can be found will be explained later.

The title of each field must be kept for later use as prompts in the entry and editing routines, and as headings for reports.

When the user defines which field is to be used as the key or identifier of a data record, this information is stored in two ways. One way is through a number which gives the key position relative to the rest of the fields (i.e., if there are 8 fields, as in Example 1, and the user selects the zip code as the key, then 6 would be the key position). The second way is through a BASIC language statement, where the variable Key (for numerics) or Key$ (for strings) is assigned to the variable name corresponding to the key field selected by the user. Consider Example 1 again. If the zip code is selected by the user for his key

field, then the assignment statement would be:

```
Key  =  N2
```

If, however, the name field is selected, the assignment statement would be:

```
Key$  =  S1$
```

If an array element is chosen, the initialization program will ask the user to specify the subscript(s) of the element he wishes to use as the key. Taking Example 1 again, if the user wants the key field to be the name of the first dependent, the key assignment statement would be:

```
Key$  =  Sal$(1)
```

Since there is the possibility of the key being either numeric or string, a number is stored in the data file which tells which type the key is. If the key is a string, the program stores a zero . If the key is a numeric field, the program stores a one.

The initialization program also stores the name of the directory file (the purpose of the directory will be explained later), the length of the key (if the key is numeric, the key length is 8 bytes - if the key is a string, then the string length is saved), and the number of records in the data file available for use (exclusive of the overhead information being discussed here).

The overhead information discussed above is stored in the following manner:

Starting at record #1, the following numbers are stored:

G - G is the record number of the head of the free list (the free list is the list of unused data records - this will be explained in detail later).

A - A is the beginning record number where the titles are stored.

B - The record number where the PRINT # statements start.

C - The record number where the READ # statements start.

D - The record number where the DIM statements start.

E - The record number where the key assignment statement starts.

F - The record number where the name of the directory file is kept.

Keytype - Keytype is Ø if the key is a string. Keytype is 1 if the key is numeric.

Keylength - The length (in bytes) of the key field.

Maxsize - The maximum number of data records available to the user.

Poskey - The position of the key field with respect to the other fields in the data records.

Field information codes - The field information codes are a series of numbers which tell how each data record of the file is formatted.

A code of 1 corresponds to a numeric field.

A code of 2 corresponds to a string field. Also, a code of 2 will be followed by a number giving the length of the string field.

A code of 3 corresponds to a numeric array. The code immediately following a 3 will give the dimensionality of the array (i.e., the number of dimensions). The dimensionality (call it d) will be followed in turn by d pairs of codes, the first member of which will be the lower subscript, and the second of which will be the upper subscript for that particular dimension.

A code of 4 corresponds to a string array. The code immediately following a 4 gives the length of each element of the array, and the code following the

string length is the dimensionality of the array. The dimensionality (call it d) will be followed in turn by d pairs of codes, the first member of which will be the lower subscript, and the second of which will be the upper subscript for that particular dimension.

A code of 0 signifies the end of the record.

If you wish to examine the contents of records 1 through G, perform the following sequence:

1. Type: SCRATCH
2. Press: EXECUTE
3. Type: 10 DIM Statement$[160], File$[6]
4. Press: STORE
5. Type: 20 END
6. Press: STORE
7. Press: RUN

The above program allocates two strings, whose use will become apparent later.

The information starting at record #1 should be read serially, as it is highly likely that the system overhead codes and the field information codes will take up more than one defined record. (The actual number of records taken up will vary according to the user's record format).

Ø. To read the information starting at record #1 serially, the following procedure would be used.
   a. Type: ASSIGN #1 to "*filename*" where *filename* is the name of the data file
   b. Press: EXECUTE
   c. Type: READ #1, 1
   d. Press: EXECUTE
   e. Type: READ #1; G,A,B,C,D,E,F, Keytype, Keylength, Maxsize, Poskey
   f. Press: EXECUTE (Now all the system overhead codes have been read in, and the serial pointer is

set at the beginning of the field information codes).

To read the field information codes, perform the following sequence:

1. Type: READ #1; X
2. Press: EXECUTE
3. Type: X
4. Press: EXECUTE

   If X is a 1, go to step 5
   If X is a 2, go to step 6
   If X is a 3, go to step 7
   If X if a 4, go to step 10
   If X is a zero, go to step 13
   If X is anything else you have made a mistake. Start over at step Ø.

5. A 1 signifies a numeric field is the next field in the data record. To find the type of the field after that, go back to step 1.
6. A 2 signifies that the next field in the data record is a string.
   a. Type: READ #1; L
   b. Press: EXECUTE
   c. Type: L
   d. Press: EXECUTE
   e. L is the length of the string field. To find out the type of the field after that, go back to step 1.
7. A 3 signifies that the next data field is a numeric array.
   a. Type: READ #1; N
   b. Press: EXECUTE
   c. Type: N
   d. Press: EXECUTE
   e. N is the number of dimensions in the numeric array.
8. Perform step 8 N times (that is, step 8 should be performed for each dimension in the array).
   a. Type: READ #1; S, T

b. Press: EXECUTE

c. Type: S, T

d. Press: EXECUTE

e. S and T are the lower and upper subscript bounds, respectively, of the ith dimension (where this is the ith time that step 8 has been performed).

9. To find the type of the next data field, go back to step 1.

10. A 4 signifies that the next data field is a string array.

a. Type: READ #1; L, N

b. Press: EXECUTE

c. Type: L, N

d. Press: EXECUTE

e. L is the length of each element in the string array, and N is the number of dimensions in the string array.

11. Perform step 11 N times (that is, step 11 should be performed for each dimension in the array).

a. Type: READ #1; S, T

b. Press: EXECUTE

c. Type: S, T

d. Press: EXECUTE

e. S and T are the lower and upper subscript bounds, respectively, of the ith dimension (where this is the ith time that step 8 has been performed).

12. To find the type of the next data field, go back to step 1.

13. A zero implies that there are no more field information codes.

Consider Example 1 again. The field information codes (which fall <u>after</u> G, A, B, C, D, E, F, Keytype, Keylength, Maxsize and Poskey) for Example 1 would be as follows: 1, 2, 30, 2, 30, 2, 30, 3, 2, 1, 3, 1, 3, 1, 4, 30, 1, 1, 7, 3, 1, 1, 7, 0.

Starting at Record #A (A is defined above):

$Title_1$ - title of the first field in the data record

$Title_2$ - title of the second field in the data record

⋮

$Title_{last}$ - title of the last field in the data record

(Note: Each title may be up to 30 characters long).

END - An EOF marker follows the last title. The End of File marker may be tested for by using the TYP function - refer to the Operating and Programming manual for details.)

Starting at Record #B (B is defined above):

PRINT # statement(s) - Note: There may be more than one PRINT # statement if the data fields do not fit on one line. Each PRINT # statement may be up to 160 characters.

END - An EOF marker follows the last PRINT # statement

Starting at Record #C (C is defined above):

READ # statement(s) (See Note with PRINT # statements)

END - EOF marker

Starting at Record #D (D is defined above):

DIM statement(s) (See Note with PRINT # statements)

END - EOF marker

Starting at Record #E (E is defined above):

Key assignment statement (The key assignment statement may be up to 160 characters long).

END - EOF marker

Starting at Record #F (F is defined above):

Filename of directory file (to be explained in detail later). The filename may be up to 6 characters long).

Starting at Record #G (G is defined above):

G + 1 - Pointer to first free record

Keysinmemory - Keysinmemory tells how many keys are in a directory file "page". This concept will be explained in detail later.

To read the information stored in records A, B, C, D, E, F, and G, perform the following instructions:

1.a. Type: READ #1,A
  b. Press: EXECUTE
2.a. Type: READ #1; Statement$
  b. Press: EXECUTE
  c. Type: Statement$
  d. Press: EXECUTE
  Statement$ will contain a title
3.a. Type: T = TYP(-1)
  b. Press: EXECUTE
  c. If T is 3, then there are no more titles. Go on to step 4 in this case.
  d. Otherwise, there are more titles. Go back to step 2.
4.a. Type: READ #1,B
  b. Press: EXECUTE
5.a. Type: READ #1; Statement$
  b. Press: EXECUTE
  c. Type: Statement$
  d. Press: EXECUTE
  Statement$ will contain a PRINT # statement
6.a. Type: T = TYP(-1)
  b. Press: EXECUTE
  c. If T is 3, then there are no more PRINT # statements. Go to step 7 in this case.
  d. Otherwise, there are more PRINT # statements. Go back to step 5.
7.a. Type: READ #1, C
  b. Press: EXECUTE
8.a. Type: READ #1; Statement$
  b. Press: EXECUTE
  c. Type: Statement$
  d. Press: EXECUTE
  Statement$ will contain a READ # statement.

9.a. Type: T = TYP(-1)

b. Press: EXECUTE

c. If T is 3, then there are no more READ # statements. Go to step 10 in this case.

d. Otherwise, there are more READ # statements. Go back to step 8.

10.a. Type: READ #1, D

b. Press: EXECUTE

11.a. Type: READ #1; Statement$

b. Press: EXECUTE

c. Type: Statement$

d. Press: EXECUTE

Statement$ will contain a DIM statement

12.a. Type: T = TYP(-1)

b. Press: EXECUTE

c. If T is 3, then there are no more DIM statements. Go on to step 13 in this case.

d. Otherwise, there are more DIM statements. Go back to step 11.

13.a. Type: READ #1, E

b. Press: EXECUTE

14.a. Type: READ #1; Statement$

b. Press: EXECUTE

c. Type: Statement$

d. Press: EXECUTE

Statement$ will contain the key assignment statement

15.a. Type: READ #1, F

b. Press: EXECUTE

16.a. Type: READ #1; File$

b. Press: EXECUTE

c. Type: File$

d. Press: EXECUTE

File$ will contain the name of the directory file.

17.a. Type: READ #1, G; H, Keysinmemory

b. Press: EXECUTE

c. Type: Keysinmemory

d. Press: EXECUTE

Keysinmemory tells how many keys (and pointers) will fit into memory.

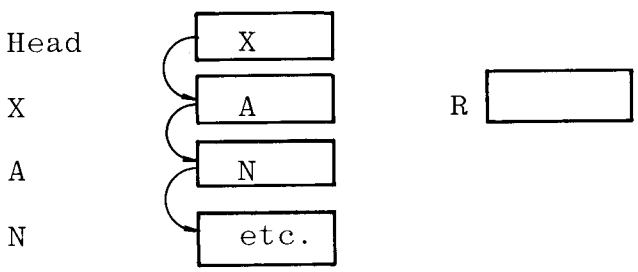
Starting at Record G+1

G+2 - Pointer to next free record

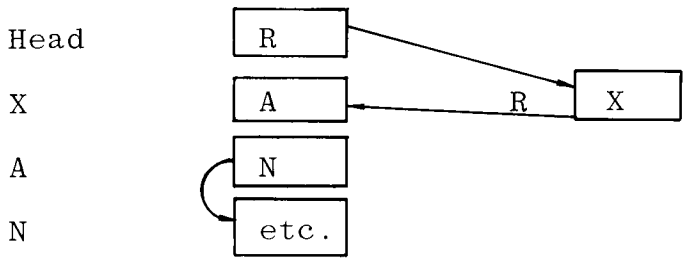
Starting at Record G+2

G+3 - Record #G, as mentioned earlier, is the head of the free list. The free list is the set of all unused records in the data file. These unused records are organized in a data structure known as a linked list. The linked list concept is simple. Each unused data record contains a number which tells where the next unused record can be found. By convention, the pointer to the first unused record is always stored in record #G. The last free record has a pointer of ∅. So, if the pointer in record #G is zero, then there are no more unused records.

Insertion and deletion from a linked list are simple also. To insert to a linked list, you merely need to store the record number of the record to be inserted in the head of the list, and store the number that was previously in the head of the list in the inserted record. Consider the following example:




Suppose you want to insert record #R into the list. The following diagram shows what the pointers in each record would be after record R is inserted:

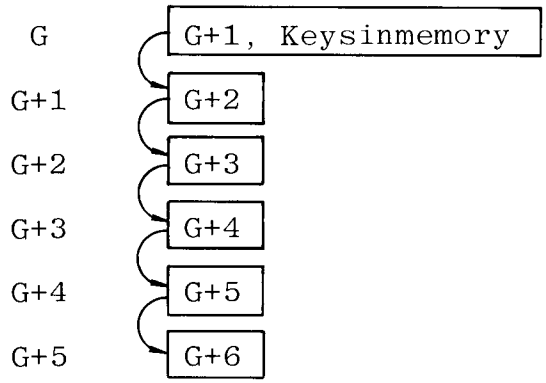



To delete a record from a linked list requires that you read the record to be deleted in order to find the pointer to the next record. Then this pointer is stored in the previous record, overwriting the pointer that was pointing to the deleted record. Consider the following example:

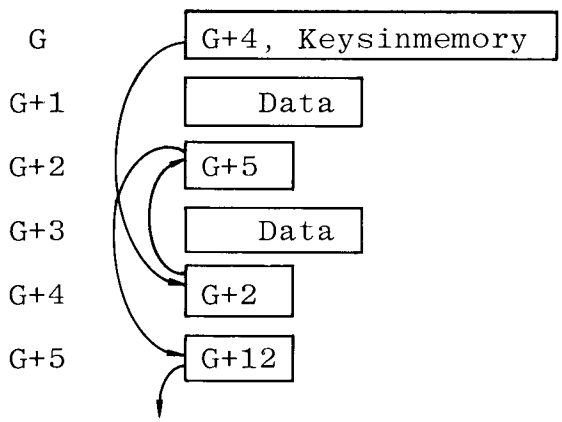

Note that for this application, insertions and deletions are always made at the beginning of the free list.

Thus, when the file is initialized, Record #G is defined to be the head of the free list, and every record after G points to the record immediately following it. Record #(G+Maxsize), the last record, has a zero. It should be noted that the sequence of the free list when it is initialized is not necessarily maintained as data records are used and discarded.
For example, when the list is initialized, it

looks like this:




After the data file has been accessed for a while, the free list might look like this:




The important factor in a linked list is to maintain the pointers correctly.

Now that the overhead information in the data file has been explained, it is necessary to explain the purpose of the directory file, which was mentioned earlier.

Since it will usually be impossible to fit the entire data base into memory at one time, some fast means of retrieving a given record from the mass storage device is needed. One way would be to keep only the key values in memory, along with a pointer associated with each key value which would tell what record number in the data file the key belonged to. If the key values were kept in ascending order, then a fast binary

search can be performed to find 1) whether or not a key exists in the data base, and 2) if so, where the record associated with that key is. This approach would work quite well, except that it imposes a restriction on the size of the data base. Many users may have data bases which are large enough that the key values and the pointers mentioned above will not fit into memory.

The next step then, is to figure out how many keys do fit into memory and then create a mass storage file whose logical records will hold the maximum number of keys that will fit into memory (along with the associated record pointers). The file should have enough logical records so that there will be room for every key (and pointer) in the data base. The problem with this approach is figuring out which record in this file (this file will henceforth be referred to as the "directory file") to look in for a given key. Obviously, it is not desirable to read every record of the directory file because that would take too long.

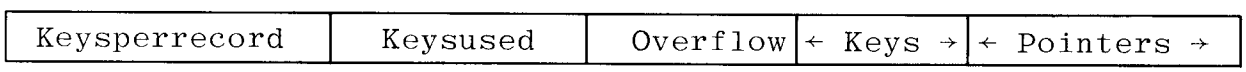
The solution to the last problem is to use a technique called "hash coding". Hashing is a method of mapping a wide range of keys to a small range of locations. Thus, we may take a key, run the key through a hashing function, and the function will come back with the record number of the directory page which should have the key and its associated pointer. The directory page is read in, a binary search is performed, and if the search is successful, we have a pointer which points to the proper data record.

Two of the primary problems with hashing are collisions and wasted space. If there is exactly enough room in a hashing table for all the probable entries, the laws of probability dictate that a lot of keys will hash to the same position in the table (refer to reference 1). Thus, most hash tables allow 40%-50% more room than will actually be needed, in the hopes that the number of collisions will be cut down. Even

in this case, however, provisions must still be made for collisions.

In this particular application, note that there are supposed to be collisions in the hashing table (each record in the directory file is supposed to hold several keys and their associated pointers). The problem to guard against here is too many collisions. In order to decrease the chances of a directory record overflowing, there are twice as many records as there need to be. In addition, each directory record has a flag which indicates whether or not that record has overflowed in the past. Thus, if the program looks in a given record for a given key, and does not find the key, then the overflow flag will be checked. If the overflow flag has been set, then the program will look in the next directory record for the key. (If the overflow flag is set in the last directory record, the program will "wrap around" and look in the first directory record.) Conversely, if a directory record is full and the program tries to insert a key in that record, the program will detect the fact that the directory record is full, set the overflow flag in that record, and then check the next directory record for room. (Again, if the last directory record is full, the "next" directory record becomes the first directory record).

The directory record format is:




Keysperrecord - The number of keys (and pointers) per directory record.

Keysused - The number of keys (and pointers) currently resident in the given directory record.

Overflow - Tells whether or not the given directory record has ever overflowed (1 for overflow, Ø for no overflow).

Keys - There will be [Keysperrecord] keys (unused keys will be set to 9 E99 if the keys are numeric and to CHR$(127) if they are strings).

Pointers - There will be [Keysperrecord] pointers (unused pointers are set to Ø.)

In most cases,the system's storage/retrieval capabilities will satisfy the user, but the user will have to write his own programs to do computations on the data. Below, an example is given to show how one might go about this.

Example 2:

Suppose the user directs the initialization program to build a data record in the following manner:

1. Numeric field (Title: Employee Number)
2. String field (Title: Name, Length: 30)
3. Numeric field (Title: Hourly Pay)
4. Numeric field (Title: Hours Worked)
5. Numeric field (Title: Pay)

Assume that someone has keyed in the first four fields for all the workers in the office into a file named "OFFICE" on a floppy disk (let 8 be the select code of the drive), and assume that the first numeric field (Employee number) is the key.

Now, the problem is two fold:

1) Compute the pay for each employee and store it in his data record, and
2) Count the total number of hours worked.

Before presenting the solution, it may be instructive to examine the contents of the files.

Assume the following data has been used:

| Name | Hourly Pay | Hours Worked | Pay | Employee Number |
|---|---|---|---|---|
| Anne Teak | 4.69 | 40 | 0.00 | 64781 |
| Archibald Barisol | 3.47 | 40 | 0.00 | 89751 |
| Ben Dover | 5.26 | 50 | 0.00 | 69885 |
| Bruce Eazely | 5.21 | 40 | 0.00 | 77845 |
| Carter Belt | 3.58 | 40 | 0.00 | 85741 |
| Jerry Atric | 3.56 | 40 | 0.00 | 37845 |
| Justin Thyme | 3.47 | 40 | 0.00 | 71213 |
| Kitten Kaboodle | 4.24 | 40 | 0.00 | 61225 |
| Kurt Remarque | 2.41 | 45 | 0.00 | 95134 |
| Larry Yet | 4.58 | 40 | 0.00 | 64784 |
| Leah Tard | 3.78 | 40 | 0.00 | 99702 |
| Marcus Absent | 3.21 | 42 | 0.00 | 75841 |
| Penny Ante | 3.88 | 25 | 0.00 | 67451 |
| Rhonda Campfire | 3.35 | 37 | 0.00 | 87541 |
| Theopholos Punnoval | 4.12 | 35 | 0.00 | 82147 |

Starting at Record #1, we find the following numbers:

```
10   - The free list's header is at record #10
 4   - The titles start at record #4
 5   - The PRINT # statement(s) start at record #5
 6   - The READ # statement(s) start at record #6
 7   - The DIM statement(s) start at record #7
 8   - The key assignment statement starts at record #8
 9   - The name of the directory is stored at record #9
 1   - The key field is numeric
 8   - There are 8 bytes in the key field
15   - There are 15 data records available
 1   - The first field is the key field
 1   - The first field in the data record is numeric
 2   - The next field in the data record is a string
30   - The string field is 30 characters long
 1 }
 1 } - The last 3 fields are all numeric
 1 }
 0   - No more fields in the record
```

Starting at record #4 are the field titles:

```
Employee Number
Name
Hourly Pay
Hours Worked
Pay
```

Starting at record #5 is the PRINT # statement by which all the data fields may be printed out to mass storage:

```
PRINT #9; N1, S1$, N2, N3, N4
```

Starting at record #6 is the READ # statement by which all the data fields may be read from the data base:

```
READ #9; N1, S1$, N2, N3, N4
```

Starting at record #7 are the DIM statements necessary to allocate space for the data fields:

```
DIM S1$[30]
DIM Key$[1]  (The last DIM statement is a dummy one,
              since the key field is numeric,not a string).
```

Starting at record #8 is the key assignement statement:

```
Key = N1  ! Employee Number
```

Starting at record #9 is the filename of the directory:

```
office
```

Starting at record #10 is the free list header:

```
  0    (This indicates that the data file is full - there
       is no more room.)
125    (This tells that there is room for 125 keys and
       pointers in each directory record.
```

Records 11 through 25 are filled with the data listed on the previous page, though, not necessarily in that order.

Now let us take a look at the directory file, "office".

```
NAME   PRO TYPE REC/FILE  BYTES/REC   ADDRESS
F8          65

office     DATA      2      2280       16/06
```

As you can see from the above catalogue, the directory has two records, each 2280 bytes long.

The first directory record has the following contents:

```
125  -  There is room for 125 keys and record pointers
        in this record.
  8  -  Of the 125, only 8 are being used at present
  0  -  This record has never overflowed
```

Next are 125 numeric keys, followed by 125 pointers.

| Keys | | | | | Pointers |
|---|---|---|---|---|---|
| 37845 | is the key of the data record | | in record | # | 13 |
| 64781 | " | " | " | " | 20 |
| 64784 | " | " | " | " | 23 |
| 67451 | " | " | " | " | 19 |
| 75841 | " | " | " | " | 25 |
| 77845 | " | " | " | " | 15 |
| 87541 | " | " | " | " | 17 |
| 89751 | " | " | " | " | 14 |

The rest of the keys are 9.0E+99, and are unused.

The rest of the pointers are Ø, and are unused.

The second directory record has the following contents:

125 - There is room for 125 keys and record pointers in this record.

7 - Of the 125, only 7 are being used at the present.

Ø - This record has never overflowed.

Next are 125 numeric keys, followed by 125 pointers.

| Keys | | | | | Pointers |
|---|---|---|---|---|---|
| 61225 | is the key of the data record | | in record | # | 12 |
| 69885 | " | " | " | " | 24 |
| 71213 | " | " | " | " | 11 |
| 82147 | " | " | " | " | 18 |
| 85741 | " | " | " | " | 21 |
| 95134 | " | " | " | " | 16 |

| | | | | |
|---|---|---|---|---|
| 99702 | " | " | " | 22 |

The rest of the keys are 9.0E+99, are unused.

The rest of the pointers are ∅, and are unused.

Armed with the knowledge of the contents of the file, it is now possible to write a program to solve the originally stated problem. An annotated listing of the solution follows:

# Solution to Example 2

```
10    MASS STORAGE IS ":F"
20    ASSIGN #9 TO "OFFICE"! Assign the file
30    READ #9,1             ! Set the serial pointer to the beginning of the file
40    READ #9;G,A,B,C,D,E,F! Read in overhead information
50    READ #9,F;Directory$ ! F gives the location of the directory file's name.
60                         !  Read the file's name.
70    ASSIGN #2 TO Directory$! Assign the directory file
80    READ #9,G;Link,Keysinmemory ! Read the arraysize (Keysinmemory) of the key
90                                !  and pointer arrays
100   CALL Compute(#9,#2,Keysinmemory,Total_hours)
110   PRINT "Total number of man-hours:";Total_hours
120   END
130   SUB Compute(#9,#2,Keysinmemory,Total_hours)
140   OPTION BASE 1
150   DIM Keys(Keysinmemory),Pointers(Keysinmemory)
160                          ! Dynamically allocate the key and pointer arrays
170   DIM S1$[30]            ! The dimension statement stored in record #D (7)
180                          !  is used to allocate space for the string field
190                          !  in the data record (Name).
200   ON END #2 GOTO Out     ! Branch out when the end of the directory file is
210                          !  found.
220   Total_hours=0          ! Initialize Total_hours.
230   Dircount=1             ! Set the directory file record pointer to the
240                          !  first directory record.
250   READ #2,Dircount;Keysinmemory,Keysused,Overflow,Keys(*),Pointers(*)
260                          ! Read in the directory record given by Dircount,
270                          !  the directory file record pointer.  (If Dircount
280                          !  is out of range, the ON END condition in line
290                          !       will cause the program to branch to the
300                          !  statement labelled Out.
310   FOR I=1 TO Keysused    ! Perform this loop for each of the used keys
320     READ #9,Pointers(I)  ! Read in each record given by the pointers in the
330                          !  current directory record
340     READ #9;N1,S1$,N2,N3,N4 ! N1 is employee number
350                          ! S1 is the name
360                          ! N2 is the hourly wage
370                          ! N3 is the number of hours
380                          ! N4 is the total pay
390     N4=N2*N3             ! Compute Pay=Rate*Hours
400     READ #9,Pointers(I)  ! Reset file pointer to beginning of data record
410     PRINT #9;N1,S1$,N2,N3,N4 ! Store the computed information (in the same
420                              !  place as it was read from.
430     Total_hours=Total_hours+N3 ! Sum up the total number of hours.
440   NEXT I
450   Dircount=Dircount+1    ! Set the directory file record pointer to read
460                          !  the next directory record.
470   GOTO 250
480 Out:   SUBEND            ! Go back to the main program.
```

Contents of File "OFFICE" After the Program on the Previous Page Has Been Executed

| Name | Hourly Pay | Hours Worked | Pay | Employee Number |
|---|---|---|---|---|
| Anne Teak | 4.69 | 40 | 187.60 | 64781 |
| Archibald Barisol | 3.47 | 40 | 138.80 | 89751 |
| Ben Dover | 5.26 | 50 | 263.00 | 69885 |
| Bruce Eazely | 5.21 | 40 | 208.40 | 77845 |
| Carter Belt | 3.58 | 40 | 143.20 | 85741 |
| Jerry Atric | 3.56 | 40 | 142.40 | 37845 |
| Justin Thyme | 3.47 | 40 | 138.80 | 71213 |
| Kitten Kaboodle | 4.24 | 40 | 169.60 | 61225 |
| Kurt Remarque | 2.41 | 45 | 108.45 | 95134 |
| Larry Yet | 4.58 | 40 | 183.20 | 64784 |
| Leah Tard | 3.78 | 40 | 151.20 | 99702 |
| Marcus Absent | 3.21 | 42 | 134.82 | 75841 |
| Penny Ante | 3.88 | 25 | 97.00 | 67451 |
| Rhonda Campfire | 3.35 | 37 | 123.95 | 87541 |
| Theopholos Punnoval | 4.12 | 35 | 144.20 | 82147 |

REFERENCES:

1. Knuth, Donald E., THE ART OF COMPUTER PROGRAMMING, VOL. 3 (SORTING AND SEARCHING) (Addison-Wesley, 1973) pp. 506-514.

2. Knuth, Donald E., THE ART OF COMPUTER PROGRAMMING, VOL. 1 (FUNDAMENTAL ALGORITHMS) (Addison-Wesley, 1973) pp. 251-256.

HEWLETT hp PACKARD

# LOAN AMORTIZATION

This program will compute and print an amortization schedule for a loan of any specified terms.

The user is required to put in the number of payments per year, plus any three of the following four catagories: The amount of each payment, the amount of money borrowed, the duration of the loan, and the annual rate of interest. The program will then compute whichever category was left unspecified and print out an amortization schedule. The amortization schedule will consist of the payment number, the amount of payment, the amount of the payment being applied toward the principle, the amount of the payment being applied to interest, and the unpaid balance of the loan. In addition, the total amount of money that the borrower will end up paying by the end of the loan period is also computed and printed.

Program Utilization:

File Name: LOAN

Variables Used:

| | | |
|---|---|---|
| Amount | - | The amount of money borrowed. |
| B | - | Temporary variable used in finding the derivative of the interest function f(i) (see the Method and Formulae section). |
| Branch | - | Variable used in conditional GOSUB to branch to the routine which calculates the unspecified input value. |
| C | - | Temporary variable used in finding the interest function f(i) (see the Method and Formulae section). |
| Count | - | Keeps track of how many lines have been printed to the CRT. |
| Flag | - | Tests to make sure that exactly one input value is left unspecified. |
| I | - | Loop counter. |
| Int | - | Shows how much of a payment applies to interest. |

| | | |
|---|---|---|
| Months | - | This variable (along with Years) helps determine the duration of the loan. |
| Noperiods | - | The total number of payment periods over the duration of the loan. |
| Payed | - | The total amount of money paid back over the duration of the loan. |
| Payment | - | Amount of each payment. |
| Payperyear | - | The number of payments in a year. |
| Rate | - | Input as the annual interest rate - changed to the interest rate per payment period. |
| S8 | - | Used as a temporary variable while solving for the interest rate. |
| Select | - | Select code of printer. |
| Tester | - | Used as a flag to see which input value is unspecified. |
| X | - | Temporary variable to hold running unpaid balance. |
| X9 | - | Temporary variable to help solve for the interest rate. |
| Y | - | Amount of each payment that is applied toward interest. |
| Years | - | This variable (along with Months) helps determine the duration of the loan. |

Special Considerations and Programming Hints:

1. This program is meant to be a stand alone problem solver rather than a subprogram.
2. System Configuration:
   Standard Memory Option
   (Optional) Printer

```
10 PRINTER IS 16
20   DIM Top$[1]
30 PRINT PAGE,"Any instructions or prompts that occur while this progr
am is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONTINUE without enter
ing"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90   Select=16
100 INPUT "Printer select code?",Select
110  Select=INT(Select)
120  IF (Select<0) OR (Select>16) THEN 90
130  Top$="N"
140  IF Select=16 THEN 240
141  PRINTER IS Select
142  PRINT
143  PRINTER IS 16
150  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
160  PRINT "has top-of-form generation and whether or not you wish to
make"
170  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
180  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
190  PRINT "generation, enter an N and press CONT.  If you press CONT
without entering"
200  PRINT "anything, Y will be the assumed response."
210  Top$="Y"
220  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
230  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"N") AND (Top$<>"n") TH
EN 210
240  PRINT PAGE;
250  PRINT "   Now you will be asked to enter values for three (out of
 four)"
260  PRINT "variables.  The four variables are:",LIN(1)
270  PRINT "           Amount of the Loan"
280  PRINT "           Annual Rate of Interest"
290  PRINT "           Duration of the Loan"
300  PRINT "           Amount of Each Payment",LIN(1)
310  PRINT "One of the categories (and only one) must be left undefi
```

```
ned."
320  PRINT "This category will be solved for by the program.  When the
 program"
330  PRINT "asks for the category you wish to solve for, simply press
CONTINUE"
340  PRINT "without entering anything  (you are required to respond to
 the "
350  PRINT "question asking for the Number of Payments in a Year).",LI
N(2)
360  Tester=6.3179664438
370  STANDARD
380 Inper:  Payment=Amount=Months=Years=Rate=Payperyear=Tester
390         Flag=0
400         INPUT "Number of Payments in a Year?",Payperyear
410         IF (Payperyear<>Tester) AND (Payperyear>0) THEN 440
420         BEEP
430         GOTO Inper
440         PRINT "Number of Payments in a Year: "&VAL$(Payperyear)
450         INPUT "Amount of the Loan?",Amount
460         IF Amount>0 THEN 520
470         BEEP
480         DISP "INVALID INPUT"
490         WAIT 500
500         Amount=Tester
510         GOTO 450
520         Ex$="Will Be Computed"
530         IF Amount<>Tester THEN Ex$=VAL$(Amount)
540         PRINT "Amount of Loan: "&Ex$
550         IF Amount<>Tester THEN B2
560         GOSUB Testflag
570         Branch=2
580 B2:   INPUT "Annual Rate of Interest?",Rate
590       IF Rate>0 THEN 650
600       BEEP
610       DISP "INVALID INPUT"
620       WAIT 500
630       Rate=Tester
640       GOTO B2
650       Ex$="Will Be Computed"
660       IF Rate<>Tester THEN Ex$=VAL$(Rate)
670       PRINT "Annual Interest Rate: "&Ex$
680       IF Rate<>Tester THEN B3
690       GOSUB Testflag
700       Branch=4
710 B3:   LINPUT "Duration of the Loan (Years,Months)",Years$
720       IF Years$<>"" THEN 750
730       Years=Months=Tester
```

```
740       GOTO 880
750       Cpos=POS(Years$,",")
760       IF Cpos=0 THEN Nomo
770       Years=VAL(Years$[1,Cpos-1])
780       Months=VAL(Years$[Cpos+1])
790       GOTO 820
800 Nomo:Months=0
810       Years=VAL(Years$)
820       IF (Years>=0) AND (Months>=0) THEN 880
830       BEEP
840       DISP "INVALID INPUT"
850       WAIT 500
860       Years=Months=Tester
870       GOTO B3
880       Ex$="Will Be Computed"
890       IF Years<>Tester THEN Ex$=VAL$(Years)&" Years, "&VAL$(Months)
&" Months"
900       PRINT "Duration of the Loan: "&Ex$
910       IF Years<>Tester THEN B4
920       GOSUB Testflag
930       Branch=3
940 B4:   INPUT "Amount of Each Payment?",Payment
950       IF Payment>0 THEN 1010
960       BEEP
970       DISP "INVALID INPUT"
980       WAIT 500
990       Payment=Tester
1000      GOTO B4
1010      Ex$="Will Be Computed"
1020      IF Payment<>Tester THEN Ex$=VAL$(Payment)
1030      PRINT "Amount of Each Payment: "&Ex$,LIN(2)
1040      IF Payment<>Tester THEN B5
1050      GOSUB Testflag
1060      Branch=1
1070 B5:  IF Flag=1 THEN B6
1080      PRINT "ERROR:  You must leave one variable unspecified."
1090      GOTO Inper
1100 B6:  IF Rate>1 THEN Rate=Rate/100
1110      Rate=Rate/Payperyear
1120      IF Branch=3 THEN 1140
1130      Noperiods=Years*Payperyear+INT(Months/12*Payperyear)+(FRACT(M
onths/12*Payperyear)<>0)
1140      ON Branch GOTO F1,F2,F3,F4
1150 F1:  Payment=Amount*Rate*(1+Rate)^Noperiods/((1+Rate)^Noperiods-1)
1160      GOTO Output
1170 F2:  Amount=Payment*((1+Rate)^Noperiods-1)/(1+Rate)^Noperiods/Rate
1180      GOTO Output
```

```
1190 F3: Noperiods=INT(-LOG(1-Amount*Rate/Payment)/LOG(1+Rate))+1
1200     GOTO Output
1210 F4: Rate=(Payment*Noperiods/Amount-1)/12
1220     IF Rate>0 THEN F41
1230     PRINT "Rate is negative or zero"
1240 END
1250 F41: Rate=1.001
1260 F42: S8=(1-Rate^(-Noperiods))/(Rate-1)
1270      C=Payment*S8
1280      X9=((Rate-1)*Noperiods*Rate^(-Noperiods)/Rate-(1-Rate^(-Nope
riods)))/(Rate-1)^2
1290      B=Payment*X9
1300      C=C-Amount
1310      Rate=Rate-C/B
1320      IF ABS(C)>.01 THEN F42
1330      Rate=Rate-1
1340      IF Payperyear*Payment>Rate*Amount*Payperyear+1 THEN Output
1350      PRINT "Your first year's payments are";Payperyear*Payment
1360      PRINT "The first year's interest is";Rate*Amount*Payperyear
1370      PRINT "Therefore, the life of the mortgage is undefined"
1380 END
1390 Output: IF Select<>16 THEN PRINT PAGE
1400         PRINTER IS Select
1410         IF (Top$="y") OR (Top$="Y") THEN PRINT PAGE;
1420         FIXED 2
1430         PRINT USING 1440;VAL$(Amount),VAL$(Rate*Payperyear*100)
1440         IMAGE "Amount of Loan: "K,/,"Annual Rate of Interest: ",K
"%"
1450         PRINT USING 1460;VAL$(Noperiods),VAL$(Noperiods/Payperyea
r)
1460         IMAGE "Duration of the Loan: ",K," Payment Periods  (",K,
" Years)"
1470         PRINT USING 1480;VAL$(Payment)
1480         IMAGE "Amount of Each Payment: ",K
1490         STANDARD
1500         PRINT USING 1510;VAL$(Payperyear)
1510         IMAGE "Number of Payments per Year: ",K,/,/
1520         IF Select<>16 THEN 1540
1530         INPUT "PRESS CONTINUE WHEN YOU ARE READY FOR THE REST OF
THE OUTPUT",Whoa
1540         PRINT "Payment #";SPA(3);"Payment";SPA(5);"Principle";SPA
(5);"Interest";SPA(3);"Unpaid Balance"
1550         X=Amount
1560         Payed=0
1570         Count=0
1580         FOR I=1 TO Noperiods
1590         Int=Rate*X
```

```
1600          Y=Payment-Int
1610          IF Y<=X THEN 1640
1620          Y=X
1630          Payment=Y+Int
1640          X=X-Y
1650          PRINT USING 1660;I,Payment,Y,Int,X
1660          IMAGE M4D,4X,M5D.2D,4X,M5D.2D,4X,M5D.2D,4X,M7D.2D
1670          Payed=Payed+Payment
1680          Count=Count+1
1690          IF (Count<>19) OR (Select<>16) THEN 1740
1700          Count=0
1710          INPUT "PRESS CONT WHEN YOU ARE READY FOR THE NEXT 20 LINE
S",Whoa
1720          PRINT PAGE;
1730          PRINT "Payment #";SPA(3);"Payment";SPA(5);"Principle";SPA
(5);"Interest";SPA(3);"Unpaid Balance"
1740          NEXT I
1750          IF Select<>16 THEN 1770
1760          INPUT "PRESS CONT WHEN YOU ARE READY FOR THE REST OF THE
OUTPUT",Whoa
1770          PRINT LIN(2);"Total Payments=";Payed;LIN(1);"Cost of Loan
=";Payed-Amount
1780          PRINT LIN(4)
1790          PRINTER IS 16
1800          INPUT "Would you like to run the program again (Y/N)?",A$
1810          IF UPC$(A$)="Y" THEN 10
1820          IF UPC$(A$)="N" THEN 1850
1830          BEEP
1840          GOTO 1800
1850          BEEP
1860          DISP "PROGRAM COMPLETED"
1870          END
1880 Testflag: IF Flag=1 THEN Errmess
1890          Flag=1
1900          RETURN
1910 Errmess: DISP "ERROR:  Only one variable may be unspecified.  Pre
ss RUN to restart."
1920          BEEP
1930          END
```

Methods and Formulae:

The method used is the amortization or direct reduction method. The borrower pays a fixed amount of money every month (or quarter, or week, depending upon the needs of the user). Part of the money pays the interest for the given period while whatever is left over is applied to the loan itself, thus leaving less capital to pay interest on in the next pay period.

$i$ = interest rate per payment period (i.e., $\frac{\text{annual interest rate}}{\text{\# payments per year}}$)

$A_0$ = original amount of loan

$n$ = # of payments needed to pay off the loan

$P$ = amount of payment

$A_1$ = amount owed after first month

$A_j$ = amount owed after j months

$$A_1 = A_0 - (P - i \cdot A_0)$$

$$A_2 = A_1 - (P - i \cdot A_1) = A_0 - (P - i \cdot A_0) - (P - i \cdot [A_0 - (P - i \cdot A_0)])$$

$$= A_0 - P + A_0 i - P + A_0 i - Pi + A_0 i^2$$

$$= A_0 (1 + 2i + i^2) - P(1+i) - P$$

$$= A_0 (1+i)^2 - P(1+i) - P$$

$$A_3 = A_2 - (P - i \cdot A_2)$$

$$= A_0 (1+i)^2 - P(1+i) - P - [P - i(A_0(1+i)^2 - P(1+i) - P)]$$

$$= A_0 (1+i)^2 - 2P - Pi - P + i(A_0(1+i)^2 - 2P - Pi)$$

$$= A_0 + 2A_0 i + A_0 i^2 - 2p - Pi - P + A_0 i + 2A_0 i^2 + A_0 i^3 - 2Pi - Pi^2$$

$$= A_0 + 3A_0 i + 3A_0 i^2 + A_0 i^3 - Pi^2 - 3Pi - 3P$$

$$= A_0(1 + 3i + 3i^2 + i^3) - P(i^2 + 2i + 1) - Pi - 2P$$

$$= A_0(1+i)^3 - P(i+1)^2 - P(i+1) - P$$

$$= A_0(1+i)^3 - P(1+i)^2 - P(1+i)^1 - P(1+i)^0$$

$$\vdots$$

$$A_j = A_0(1+i)^j - P \sum_{k=0}^{j-1} (1+i)^k$$

$$An\ (=0) = A_0(1+i)^n - P \sum_{k=0}^{n-1} (1+i)^k$$

Now, using the identity

$$\sum_{j=0}^{n-1} (1+i)^j = \frac{(1+i)^n-1}{i}$$

(Proof: let $n = 3$

Then $\sum_{j=0}^{2} (1+i)^j = \frac{(1+i)^3-1}{i}$

$$1+(1+i)+1+2i+i^2) = \frac{1+3i+3i^2+i^3-1}{i}$$

$$3+3i+i^2 = 3+3i+i^2$$

so the identity is proven for n=3.
Now assume that the identity is true for n.
If we can prove it true for n+1 then by induction, the identity holds for all n.

$$\sum_{j=0}^{n-1} (i+i)^j = \frac{(1+i)^n-1}{i}$$

$$\sum_{j=0}^{n-1} (1+i)^j + (1+i)^n = \frac{(1+i)^n-1}{i} + (1+i)^n$$

$$\sum_{j=0}^{n} (1+i)^j = \frac{(1+i)^n-1}{i} + \frac{i(1+i)^n}{i}$$

$$= \frac{(1+i)^n + i(1+i)^n-1}{i}$$

$$= \frac{(1+i)^n(1+i)-1}{i}$$

$$= \frac{(1+i)^{n+1}-1}{i}$$

Q. E. D)

we may substitute for the sumnation to get the formula:

$$\emptyset = A_0(1+i)^n - P\,\frac{(1+i)^n-1}{i}$$

solving for $A_0$, P, and n gives:

$$A_0 = P\,\frac{(1+i)^n-1}{i(1+i)^n}$$

$$P = \frac{A_0\, i(1+i)^n}{(1+i)^n-1}$$

$$\emptyset = P(1+i)^n - P - A_0 i(1+i)^n$$

$$\emptyset = (1+i)^n\,(P-A_0 i) - P$$

$$(1+i)^n = \frac{P}{P-A_0 i}$$

$$\log_{(1+i)}(1+i)^n = \log_{(1+i)}\left(\frac{P}{P-A_0 i}\right)$$

$$n = \log_{(1+i)}\left(\frac{P}{P-A_0 i}\right)$$

and since $\log_a x = \frac{\log_b x}{\log_b a}$)

$$n = \frac{\ln\left(\frac{P}{P-A_0 i}\right)}{\ln(1+i)}$$

To solve for i requires an iterative rootfinder. Here, a Newton-Raphson method is used:

$$i_{j+1} = i_j - \frac{f(i_j)}{f'(i_j)}$$

where $f(i) = (1+i)^n [P-A_0 i] - P$

and $f'(i) = n(1+i)^{n-1} [P-A_0 i] - A_0(1+i)^n$

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "LOAN: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.
      2) If you do not want page feeds:
         a) Enter: N

b) Press: CONT

c) Go to step 5.

5. When "Number of Payments in a Year?" appears in the display area:

   a. Enter: The number of payments you will make in a year.

   b. Press: CONT

6. For any one of the following questions, the user may press CONT without entering anything in response to the prompt.

   a. When "Amount of the Loan?" appears in the display area:

      1) Enter: The total amount of money borrowed.

      2) Press: CONT

   b. When "Annual Rate of Interest?" appears in the display area:

      1) Enter: The annual interest rate.

      2) Press: CONT

      Note: The interest rate may either be entered in hundredths or in percentages. Thus, 9.5 and .095 are equivalent responses.

   c. When "Duration of the Loan (Years, Months)?" appears in the display area:

      1) Enter: The number of years, followed by a comma, followed by the number of months, that defined the loan period.

      2) Press: CONT

   d. When "Amount of Each Payment?" appears in the display area:

      1) Enter: The amount of money to be paid for each payment period.

      2) Press: CONT

7. If you entered 16 for the printer select code in part 4a, only 20 lines of output will be displayed on the CRT. Press CONT whenever you are ready to view the next 20 lines.

8. When "Would you like to run the program again (Y/N)?" appears in the display area:

   a. If you would like to run the program again:

      1) Type: Y

      2) Press: CONT

      3) Go to step 4.

b. If you would not like to run the program again:

1) Type: N

2) Press: CONT

3) The program stops.

## EXAMPLE

```
Amount of Loan: 10000.00
Annual Rate of Interest: 8.50%
Duration of the Loan: 36.00 Payment Periods  (3.00 Years)
Amount of Each Payment: 315.68
Number of Payments per Year: 12
```

| Payment # | Payment | Principle | Interest | Unpaid Balance |
|---|---|---|---|---|
| 1 | 315.68 | 244.84 | 70.83 | 9755.16 |
| 2 | 315.68 | 246.58 | 69.10 | 9508.58 |
| 3 | 315,68 | 248.32 | 67.35 | 9260.26 |
| 4 | 315.68 | 250.08 | 65.59 | 9010.18 |
| 5 | 315.68 | 251.85 | 63.82 | 8758.32 |
| 6 | 315.68 | 253.64 | 62.04 | 8504.69 |
| 7 | 315.68 | 255.43 | 60.24 | 8249.25 |
| 8 | 315.68 | 257.24 | 58.43 | 7992.01 |
| 9 | 315.68 | 259.07 | 56.61 | 7732.94 |
| 10 | 315.68 | 260.90 | 54.78 | 7472.04 |
| 11 | 315.68 | 262.75 | 52.93 | 7209.30 |
| 12 | 315.68 | 264.61 | 51.07 | 6944.69 |
| 13 | 315.68 | 266.48 | 49.19 | 6678.20 |
| 14 | 315.68 | 268.37 | 47.30 | 6409.83 |
| 15 | 315.68 | 270.27 | 45.40 | 6139.56 |
| 16 | 315.68 | 272.19 | 43.49 | 5867.37 |
| 17 | 315.68 | 274.11 | 41.56 | 5593.26 |
| 18 | 315.68 | 276.06 | 39.62 | 5317.20 |
| 19 | 315.68 | 278.01 | 37.66 | 5039.19 |
| 20 | 315.68 | 279.98 | 35.69 | 4759.21 |
| 21 | 315.68 | 281.96 | 33.71 | 4477.24 |
| 22 | 315.68 | 283.96 | 31.71 | 4193.28 |
| 23 | 315.68 | 285.97 | 29.70 | 3907.31 |
| 24 | 315.68 | 288.00 | 27.68 | 3619.31 |
| 25 | 315.68 | 290.04 | 25.64 | 3329.27 |
| 26 | 315.68 | 292.09 | 23.58 | 3037.18 |
| 27 | 315.68 | 294.16 | 21.51 | 2743.02 |
| 28 | 315.68 | 296.25 | 19.43 | 2446.77 |
| 29 | 315.68 | 298.34 | 17.33 | 2148.43 |
| 30 | 315.68 | 300.46 | 15.22 | 1847.97 |
| 31 | 315.68 | 302.59 | 13.09 | 1545.38 |
| 32 | 315.68 | 304.73 | 10.95 | 1240.65 |
| 33 | 315.68 | 306.89 | 8.79 | 933.77 |
| 34 | 315.68 | 309.06 | 6.61 | 624.71 |
| 35 | 315.68 | 311.25 | 4.42 | 313.46 |
| 36 | 315.68 | 313.46 | 2.22 | 0.00 |

```
Total Payments= 11364.313472
Cost of Loan= 1364.313472
```

HEWLETT hp PACKARD

# SAVINGS ACCOUNT/COMPOUND INTEREST ANALYSIS

This program will compute and print the terms of a savings account under compound interest.

The user is required to input the number of times per year that the interest is compounded, plus any three of the following four categories: The original principle, the duration of the account, the principle in the account at the end of the duration of the account, and the annual rate of interest.

The program will then compute whichever category was left unspecified and print a table showing the amount of interest added to the account after each time that the interest is compounded, plus the old and new balances at the end of every compounding period.

Program Utilization:

File Name: SAVACC

Variables:

| | | |
|---|---|---|
| Branch | - | tells which category is undefined (may range from 1 to 4). |
| Count | - | counts how many lines of output are printed at a time (used if printing to a CRT). |
| Flag | - | indicates whether or not a field has been left unspecified. |
| I | - | loop counter. |
| Int | - | the amount of interest earned in a single compounding period. |
| Months | - | the number of months the account is in effect. |
| Nocompounds | - | The number of compounding periods in a single year. |
| Noperiods | - | The number of compounding periods in the life of the account. |
| Principle | - | The original amount of money in the account. |

| | | |
|---|---|---|
| Rate | - | The rate of interest for a single compounding period. |
| Select | - | The select code of the printer (16 for the CRT). |
| Tester | - | Tests to see if a value is entered for each category. |
| Top$ | - | Tells whether or not to use paging on a hardcopy printer. |
| X | - | temporary variable for old balance. |
| Y | - | temporary variable for new balance. |
| Years | - | the number of years the account is in effect. |

Special Considerations:

1. This program is meant to be a stand alone problem solver, rather than a subprogram.
2. System Configuration:
   Standard Memory Option
   (Optional) Printer

## Listing of Savings Account/Compound Interest Analyzer

```
10   PRINTER IS 16
20   DIM Top$[1]
30   PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONTINUE without enter
ing"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90   Select=16
100 INPUT "Printer select code?",Select
110  Select=INT(Select)
120  IF (Select<0) OR (Select>16) THEN 90
130  PRINTER IS Select
140  PRINT
150  PRINTER IS 16
160  Top$="N"
170  IF Select=16 THEN 270
171  PRINTER IS Select
172  PRINT
173  PRINTER IS 16
180  PRINT PAGE;"Please indicate below whether or not the printer you
are using"
190  PRINT "has top-of-form generation and whether or not you wish to
make"
200  PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
210  PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
220  PRINT "generation, enter an N and press CONT.  If you press CONT
without entering"
230  PRINT "anything, Y will be the assumed response."
240  Top$="Y"
250  INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
260  IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 240
270  PRINT PAGE;
280  PRINT "   Now you will be asked to enter values for three (out of
 four)"
290  PRINT "variables.  The four variables are:",LIN(1)
300  PRINT "          Amount of Original Principle"
310  PRINT "          Annual Rate of Interest"
```

```
320  PRINT "           Duration of the Account"
330  PRINT "           Amount of Principle at Termination of the Accoun
t",LIN(1)
340  PRINT "One of the categories (and only one) must be left undefi
ned."
350  PRINT "This category will be solved for by the program.  When the
 program"
360  PRINT "asks for the category you wish to solve for, simply press
CONT"
370  PRINT "without entering anything  (you are required to respond
to the "
380  PRINT "question asking for the Number of Compoundings in a Year).
",LIN(1)
390  Tester=6.3179664438
400  STANDARD
410 Inper:  Principle=Total=Months=Years=Rate=Nocompounds=Tester
420         Flag=0
430         INPUT "Number of Compoundings in a Year?",Nocompounds
440         IF (Nocompounds<>Tester) AND (Nocompounds>0) THEN 470
450         BEEP
460         GOTO Inper
470         PRINT "Number of Compoundings in a Year: "&VAL$(Nocompound
s)
480         INPUT "Amount of Original Principle?",Principle
490         IF Principle>0 THEN 550
500         BEEP
510         DISP "INVALID INPUT"
520         WAIT 500
530         Principle=Tester
540         GOTO 480
550         Ex$="Will Be Computed"
560         IF Principle<>Tester THEN Ex$=VAL$(Principle)
570         PRINT "Principle: "&Ex$
580         IF Principle<>Tester THEN B2
590         GOSUB Testflag
600         Branch=1
610 B2:   INPUT "Annual Rate of Interest?",Rate
620       IF Rate>0 THEN 680
630       BEEP
640       DISP "INVALID INPUT"
650       WAIT 500
660       Rate=Tester
670       GOTO B3
680       Ex$="Will Be Computed"
690       IF Rate<>Tester THEN Ex$=VAL$(Rate)
700       PRINT "Rate: "&Ex$
710       IF Rate<>Tester THEN B3
```

```
720       GOSUB Testflag
730       Branch=4
740 B3:   LINPUT "Duration of the Account (Years,Months)",Years$
750       IF Years$<>"" THEN 780
760       Years=Months=Tester
770       GOTO 910
780       Cpos=POS(Years$,",")
790       IF Cpos=0 THEN Nomo
800       Years=VAL(Years$[1,Cpos-1])
810       Months=VAL(Years$[Cpos+1])
820       GOTO 850
830 Nomo:Months=0
840       Years=VAL(Years$)
850       IF (Years>=0) AND (Months>=0) THEN 910
860       BEEP
870       DISP "INVALID INPUT"
880       WAIT 500
890       Years=Months=Tester
900       GOTO B3
910       Ex$="Will Be Computed"
920       IF Years<>Tester THEN Ex$=VAL$(Years)&" Years, "&VAL$(Months)
&" Months"
930       PRINT "Duration: "&Ex$
940       IF Years<>Tester THEN B4
950       GOSUB Testflag
960       Branch=3
970 B4:   INPUT "Amount of the Principle at Termination of the Account?
",Total
980       IF Total>0 THEN 1040
990       BEEP
1000      DISP "INVALID INPUT"
1010      WAIT 500
1020      Total=Tester
1030      GOTO B5
1040      Ex$="Will Be Computed"
1050      IF Total<>Tester THEN Ex$=VAL$(Total)
1060      PRINT "Amount at Termination: "&Ex$,LIN(2)
1070      IF Total<>Tester THEN B5
1080      GOSUB Testflag
1090      Branch=2
1100 B5:  IF Flag=1 THEN B6
1110      PRINT "ERROR:  You must leave one variable unspecified."
1120      GOTO Inper
1130 B6:  IF Rate>=1 THEN Rate=Rate/100
1140      Rate=Rate/Nocompounds
1150      IF Branch=3 THEN 1170
1160      Noperiods=Years*Nocompounds+INT(Months/12*Nocompounds)+(FRACT
```

```
(Months/12*Nocompounds)<>0)
1170      IF Branch<3 THEN 1250
1180      IF Total>=Principle THEN 1250
1190      BEEP
1200      DISP "ERROR"
1210      PRINT LIN(1),"The amount of principle at termination of the a
ccount can not"
1220      PRINT "be less than the original amount of principle.  Start
over.",LIN(1)
1230      WAIT 700
1240      GOTO Inper
1250      ON Branch GOTO F1,F2,F3,F4
1260 F1: Principle=Total*(1+Rate)^(-Noperiods)
1270      GOTO Output
1280 F2: Total=Principle*(1+Rate)^Noperiods
1290      GOTO Output
1300 F3: Noperiods=LOG(Total/Principle)/LOG(1+Rate)
1310      GOTO Output
1320 F4: Rate=(Total/Principle)^(1/Noperiods)-1
1330 Output: IF Select<>16 THEN PRINT PAGE
1340         PRINTER IS Select
1350         IF (Top$="y") OR (Top$="Y") THEN PRINT PAGE;
1360         FIXED 2
1370         PRINT USING 1380;VAL$(Principle),VAL$(Rate*Nocompounds*10
0)
1380         IMAGE "Amount of Original Principle: "K"",/,"Annual Rate
of Interest: ",K"%"
1390         PRINT USING 1400;VAL$(Noperiods),VAL$(Noperiods/Nocompoun
ds)
1400         IMAGE "Duration of the Account: ",K," Compounding periods
  (",K," Years)"
1410         PRINT USING 1420;VAL$(Total)
1420         IMAGE "Amount of Principle at Termination of the Account:
 ",K
1430         STANDARD
1440         PRINT USING 1450;Nocompounds
1450         IMAGE "Number of Compoundings per Year: ",K,/,/
1460         IF Select<>16 THEN 1480
1470         INPUT "PRESS CONTINUE WHEN YOU ARE READY FOR THE REST OF
THE OUTPUT",Whoa
1480         PRINT " Period #";SPA(5);"Old Balance          Interest";S
PA(7);"New Balance"
1490         X=Principle
1500         Count=0
1510         FOR I=1 TO Noperiods
1520         Int=Rate*X
1530         Y=X+Int
```

```
1540          PRINT USING 1550;I,X,Int,Y
1550          IMAGE M5D,7X,M7D.2D,7X,M6D.2D,7X,M7D.2D
1560          X=Y
1570          Count=Count+1
1580          IF (Count<>19) OR (Select<>16) THEN 1630
1590          Count=0
1600          INPUT "PRESS CONTINUE WHEN YOU ARE READY FOR THE NEXT 20
LINES",Whoa
1610          PRINT PAGE;
1620          PRINT " Period #";SPA(5);"Old Balance          Interest";S
PA(7);"New Balance"
1630          NEXT I
1640          IF Select<>16 THEN 1660
1650          INPUT "PRESS CONTINUE WHEN YOU ARE READY FOR THE REST OF
THE OUTPUT",Whoa
1660          PRINT LIN(2);"Original Principle: ";Principle;LIN(1);"End
ing Principle: ";Total;LIN(1);"Profit: ";Total-Principle
1670          PRINT LIN(4)
1680          PRINTER IS 16
1690          INPUT "Would you like to run the program again (Y/N)?",A$
1700          IF UPC$(A$)="Y" THEN 10
1710          IF UPC$(A$)="N" THEN 1740
1720          BEEP
1730          GOTO 1690
1740          BEEP
1750          DISP "PROGRAM COMPLETED"
1760          END
1770 Testflag: IF Flag=1 THEN Errmess
1780          Flag=1
1790          RETURN
1800 Errmess: DISP "ERROR:  Only one variable may be unspecified.  Pre
ss RUN to restart."
1810 END
```

Method and Formulae:

let $x$ = # of compoundings in a year

$i$ = annual rate of interest

$P_0$ = original principle

$P_j$ = principle after j compounding periods

$$P_1 = P_0 \left(1 + \frac{i}{x}\right)$$

$$P_2 = P_1 \left(1 + \frac{i}{x}\right) = P_0 \left(1 + \frac{i}{x}\right)\left(1 + \frac{i}{x}\right) = P_0 \left(1 + \frac{i}{x}\right)^2$$

$$P_3 = P_0 \left(1 + \frac{i}{x}\right)^3$$

$$\vdots$$

$$P_j = P_0 \left(1 + \frac{i}{x}\right)^j$$

Now if n = the number of years that the principle is left at interest, then there are nx compounding periods, and the principle at the end of n years is

$$P_{nx} = P_0 \left(1 + \frac{i}{x}\right)^{nx}$$

Solving for $P_0$ gives:

$$P_0 = P_{nx} \left(1 + \frac{i}{x}\right)^{-nx}$$

solving for nx gives

$$\left(1 + \frac{i}{x}\right)^{nx} = \frac{P_{nx}}{P_0}$$

$$\log_{\left(1+\frac{i}{x}\right)} \left(1 + \frac{i}{x}\right)^{nx} = \log_{\left(1+\frac{i}{x}\right)} \left(\frac{P_{nx}}{P_0}\right)$$

$$nx = \log (P_{nx}/P_0) / \log \left(1 + \frac{i}{x}\right)$$

and solving for i gives

$$\left(1 + \frac{i}{x}\right)^{nx} = \frac{P_{nx}}{P_0}$$

$$1 + \frac{i}{x} = \left(\frac{P_{nx}}{P_0}\right)^{1/nx}$$

$$\frac{i}{x} = \left(\frac{P_{nx}}{P_0}\right)^{1/nx} - 1$$

$$i = x\left[\left(\frac{P_{nx}}{P_0}\right)^{1/nx} - 1\right]$$

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "SAVACC: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c. Go to step 5.
5. When "Number of Compoundings in a Year?" appears in the display area:
   a. Enter: The number of times that the interest on your money will be compounded in a year.
   b. Press: CONT
6. For any <u>one</u> of the following questions, the user may press CONT without entering anything in response to the prompt.
   a. When "Amount of Original Principle?" appears in the display area:
      1) Enter: The amount of original principle.

2) Press: CONT

b. When "Annual Rate of Interest?" appears in the display area:

1) Enter: The annual rate of interest. (Note: The interest rate may be entered either as a percentage or as a decimal. Any value less than 1 will be treated as a decimal. Any value greater than or equal to 1 will be treated as a percentage.)

2) Press: CONT

c. When "Duration of the Account (Years, Months)?" appears in the display area:

1) Enter: The length of time for which you want to leave the money in the account (in terms of years and months - separate the two with a comma).

2) Press: CONT

d. When "Amount of the Principle of Termination of the Account?" appears in the display area:

1) Enter: The amount of money in the account at the time of termination.

2) Press: CONT

7. If you entered 16 for the printer select code in part 4a, only 20 lines of output at a time will be displayed on the CRT. Press CONT whenever you are ready to view the next 20 lines.

8. When "Would you like to run the program again (Y/N)?" appears in the display area:

a. If you would like to run the program again:

1) Type: Y

2) Press: CONT

3) Go to step 4.

or

a. If you do not want to run the program again:

1) Type: N

2) Press: CONT

3) The program stops.

## EXAMPLE

```
Amount of Original Principle: 10000.00
Annual Rate of Interest: 6.00%
Duration of the Account: 4.00 Compounding periods  (1.00 Years)
Amount of Principle at Termination of the Account: 10613.64
Number of Compoundings per Year: 4


 Period #      Old Balance          Interest         New Balance
     1           10000.00            150.00           10150.00
     2           10150.00            152.25           10302.25
     3           10302.25            154.53           10456.78
     4           10456.78            156.85           10613.64


Original Principle:  10000
Ending Principle:  10613.6355063
Profit:  613.6355063
```

HEWLETT hp PACKARD

# HOUSEHOLD BUDGET ANALYZER

This program helps plan and record a person's or family's household budget. The program will hold data for a complete year at any one given time.

The program allows the user to define the categories on which he spends money, edit those categories, enter his budget forecast, edit that forecast, enter his expenditures, edit those expenditures, review past budgets, reconcile forecasts with expenditures, and add up year-to-date expenditures.

There are 13 general categories for expenses and one for income. Each category may be subdivided into as many subcategories as the user desires. The categories are:

Shelter
Food and Groceries
Automobile Expenses
Clothing
Entertainment
Vices
Contributions
Insurance Premiums
Medical Bills
Gifts
Future Investments
Installment Payments
Miscellaneous
Income

Program Utilization:

File Name: BUDGET

Variables:

A$ - Used for user responses to questions asked by the program.

Budget $(*) - This string array holds the names of the budget categories and subcategories.

Budget(*) - This SHORT precision four-dimensional array holds the data. The first dimension specifies which budget category, the second dimension specifies the subcategory, the third dimension distinguishes between predicted and actual expenditures, and the fourth dimension gives the month of the year.

Check - Return variable on ASSIGN statement telling whether or not a file has been initialized.

File$ - Holds the name of the file being used.

Flag$ - Used to test whether or not the user entered anything in response to a question.

Flagflag - Tells the program whether or not a message needs to be issued.

I - Loop counter.

Incswitch - Tells the program whether to add or subtract a number (distinguishes between income and expenditures in report).

Index(*) - Tells how many subcategories are in each category.

Input$ - Accepts user responses.

Insert$ - A phrase which makes one prompt message serve two different purposes.

J - Loop counter.

Lenfil - The length of the file name being used to store data.

Mo - The number of the month being used.

Mode$ - Either "PREDICTED" or "ACTUAL" depending upon whether forecasts or expenditures are being entered.

| | | |
|---|---|---|
| Model | - | Tells whether forecasts or expenditures are being entered. |
| Months$(*) | - | Array holding the names of the months. |
| Pos | - | The position of a letter in an indexing array. |
| Select | - | The select code of the printer. |
| Subindex$ | - | Holds the small letters a through j. |
| Subindexc$ | - | Holds the capital letters A through J. |
| Totalout | - | Temporary variable for adding data together. |

Special Considerations and Programming Hints:

1. The select code of the printer may be changed at any time by typing in "Select = n" where n is the select code (Note: The select code of the CRT is 16, and the select code of the internal printer is Ø, if your machine has an internal printer.)
2. If there are no subcategories under a certain category, you will not be allowed to enter any amounts under that category.
3. Everything under the category "Future Investments" is subtracted from "Income" the same as the other categories. Even though you may still have control of the money (i.e., stock or savings), it is money which you have set aside <u>not</u> to spend on something else.
4. System Configuration
   Standard Memory Option
   (Optional) Printer

## Listing of Program Budget

```
10 OPTION BASE 1
20 SERIAL
30 DIM Budget#(14,11)[20],Flag#[20],Input#[20],Subindex#[10],Subindexc
#[10]
40 DIM File#[10],Mode#[10],Months#(12)[9],C#[1],Insert#[18]
50 INTEGER Index(14)
60 SHORT Budget(14,11,2,12)
70  MAT READ Months#
80  DATA January,February,March,April,May,June,July, August,September,
October,November,December
90  Flag#="MXYZPTLK"
100 Subindex#="abcdefghij"
110 Subindexc#="ABCDEFGHIJ"
120 PRINTER IS 16
130  PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is "
140  PRINT "running will appear on the CRT only.  The output for the p
rogram"
150  PRINT "will be printed on the device having the select code you a
re"
160  PRINT "asked to enter below.  If you press CONTINUE without enter
ing"
170  PRINT "a number, the output will appear on the default printer, t
he CRT"
180  PRINT "(select code 16).",LIN(4)
190  Select=16
200  INPUT "Printer select code?",Select
210  Select=INT(Select)
220  IF (Select<0) OR (Select>16) THEN 190
221  PRINTER IS Select
222  PRINT
223  PRINTER IS 16
230  PRINT PAGE;"   Please enter the name of the file containing your
data.  If"
240  PRINT "this is the first time you have run the program, the progr
am will"
250  PRINT "build the file for you, naming it whatever you wish.  It i
s important"
260  PRINT "that you let the program create the file, because there ar
e certain"
270  PRINT "things which must be done to initialize it properly.  File
 names can"
280  PRINT "be up to six characters long."
290  INPUT "Please enter the filename",File$
300  Lenfil=LEN(File$)
310  IF Lenfil>6 THEN 290
320  File$[Lenfil+1]=":"
```

```
330  PRINT PAGE;"   The data for this program will be stored on a file
 called"
340  PRINT File$[1,Lenfil];".  You may select the device you wish to u
se to store the data"
350  PRINT "on (i.e., tape cartridge, floppy disk, hard disk, etc.).
To specify"
360  PRINT "the desired device, type its one-letter device code, follo
wed"
370  PRINT "immediately by a two-digit select code (i.e., T15, F8, D10
, etc.),"
380  PRINT "and press CONTINUE.  If you press CONTINUE without enterin
g anything,"
390  PRINT "T15 will be the default storage medium."
400  File$[Lenfil+2]="T15"
410  INPUT "Please enter the device code of your storage medium",File$
[Lenfil+2]
420  DISP "Is ";File$;" okay (Y/N)";
430  INPUT A$
440  IF (A$="Y") OR (A$="y") THEN 470
450  IF (A$="N") OR (A$="n") THEN 400
460  GOTO 420
470  ASSIGN #1 TO File$,Check
480  IF Check=1 THEN 840
490  PRINT PAGE;" Please wait until the data has been read in."
500  DISP " Please wait until the data has been read in."
510  GOSUB Readchr
520  GOSUB Readnum
530  GOSUB Prompt
540  GOTO 700



**********************


560 Prompt:  IF Select<>16 THEN PRINT PAGE;
570  PRINT LIN(2),"To run the program, press one of the user defined k
eys (UDK's)."
580  PRINT "The UDK's are defined in the following manner:"
590  PRINT SPA(10);"k0  -- Initialize the budget categories"
600  PRINT SPA(10);"k1  -- Forecast the monthly budget"
610  PRINT SPA(10);"k2  -- Enter the monthly expenditures"
620  PRINT SPA(10);"k3  -- Reconcile the forecast with the expenditure
s"
630  PRINT SPA(10);"k4  -- Review past budgets"
640  PRINT SPA(10);"k8  -- Edit the budget categories"
```

```
650  PRINT SPA(10);"k9  -- Edit the monthly forecast"
660  PRINT SPA(10);"k10 -- Edit the monthly expenditures"
670  PRINT SPA(10);"K11 -- Add up actual year-to-date expenditures"
680  PRINT SPA(10);"k15 -- End the program"
690  RETURN
700  ON KEY #0 GOSUB Initialize
710  ON KEY #1 GOSUB Forecast
720  ON KEY #2 GOSUB Endmonth
730  ON KEY #3 GOSUB Reconcile
740  ON KEY #4 GOSUB Review
750  ON KEY #8 GOSUB Editinit
760  ON KEY #9 GOSUB Editfore
770  ON KEY #10 GOSUB Editexpen
780  ON KEY #11 GOSUB Year
790  ON KEY #15 GOSUB Store
800  Flagflag=1
810  DISP "Select a Key"
820  GOTO 810


**********************


840  PRINT PAGE;"   Please wait while your data file is being initiali
zed."
850  PRINT "You will receive further instructions when everything is r
eady.",LIN(2)
860  CREATE File$,72
870  ASSIGN #1 TO File$
880  FOR I=1 TO 14
890     READ Budget$(I,1)
900  NEXT I
910  DATA Shelter,Food and Groceries,Automobile Expenses,Clothing
920  DATA Entertainment,Vices,Contributions,Insurance Premiums,Medical
 Bills
930  DATA Gifts,Future Investments,Installment Payments,Miscellaneous,
Income
940  GOSUB Initialize
950  READ #1,14
960  MAT PRINT #1;Budget
970  GOSUB Prompt
980  GOTO 700
```

```
************************


1000 Initialize:PRINTER IS Select
1010 IF Select=16 THEN PRINT PAGE;
1020 PRINT "As each main category is displayed, you are asked"
1030 PRINT "to enter as many subcategories as you want.   When you hav
e entered"
1040 PRINT "all the subcategories that you want, merely press CONT wit
hout"
1050 PRINT "entering anything.  The main categories are given below."
1060 FOR I=1 TO 14
1070    PRINT TAB(10);Budget$(I,1)
1080 NEXT I
1090 MAT Index=ZER
1100 PRINT
1110 PRINTER IS 16
1120 Input$=Flag$
1130 Loop1:FOR I=1 TO 14
1140 PRINT Budget$(I,1)
1150 Enter1: DISP "CATEGORY:";Budget$(I,1);"--------Please enter subca
tegory ";Subindex$[Index(I)+1,Index(I)+1]&".";
1160    INPUT " ",Input$
1170    IF Input$=Flag$ THEN Endloop1
1180    Index(I)=Index(I)+1
1190    Budget$(I,Index(I)+1)=Input$
1200    PRINT TAB(5);Subindex$[Index(I),Index(I)];".  ";Budget$(I,1+In
dex(I))
1210    Input$=Flag$
1220    IF Index(I)<=9 THEN Enter1
1230    DISP "Only 10 subcategories are allowed--proceeding to the nex
t category"
1240    BEEP
1250    WAIT 5000
1260 Endloop1: PRINT
1270    NEXT I
1280    GOSUB Formprint
```

```
************************


1300 Editinit: INPUT "Do you wish to make any changes in the budget ca
tegories (Y/N)?",A$
1310            IF (A$="Y") OR (A$="y") THEN Change
1320            IF (A$="N") OR (A$="n") THEN Nochange
1330            GOTO 1300
1340 Change:  PRINT PAGE;"Instructions for changing subcategories:"
1350 PRINT "As each main category is displayed, you are given the oppo
rtunity"
1360 PRINT "to change any of its subcategories.  If you enter N when a
sked if "
1370 PRINT "you want to make any changes under a certain category, the
 program"
1380 PRINT "will proceed to the next one.  If you enter Y, then each s
ubcategory"
1390 PRINT "will be displayed one at a time.  You may then either pres
s CONT"
1400 PRINT "without typing anything, thus leaving the subcategory unch
anged, or"
1410 PRINT "you may type in a different subcategory before pressing CO
NT."
1420 PRINT "After every main category has been gone through, you will
be given"
1430 PRINT "the opportunity to make additions and deletions."
1440 Loop2: FOR I=1 TO 14
1450            PRINT LIN(1);Budget$(I,1)
1460            FOR J=1 TO Index(I)
1470               PRINT TAB(5);Subindex$[J,J];".  ";Budget$(I,J+1)
1480            NEXT J
1490 Ask: A$="N"
1500      DISP "Do you wish to make any changes under the category ";B
udget$(I,1);" (Y/N)";
1510      INPUT A$
1520      IF (A$="N") OR (A$="n") THEN Endloop2
1530      IF (A$="Y") OR (A$="y") THEN Edit
1540      GOTO Ask
1550 Edit: FOR J=1 TO Index(I)
1560          EDIT "Make any changes you want and press CONT",Budget$(
I,1+J)
1570       NEXT J
1580       A$="N"
```

```
1590        INPUT "Do you wish to make any additions or deletions (Y/N)
?",A$
1600        IF (A$="N") OR (A$="n") THEN Endloop2
1610        IF (A$="Y") OR (A$="y") THEN Addel
1620        GOTO 1590
1630 Addel: INPUT "Do you wish to add or delete (A/D)?",A$
1640         IF (A$="a") OR (A$="A") THEN Add
1650         IF (A$="D") OR (A$="d") THEN Delete
1660         GOTO Addel


************************


1680 Add: IF Index(I)>=10 THEN Endloop2
1690      DISP "CATEGORY:";Budget$(I,1);"--------Please enter subcateg
ory ";Subindex$[Index(I)+1,Index(I)+1]&".";
1700      INPUT " ",Input$
1710      IF Input$=Flag$ THEN 1580
1720      Index(I)=Index(I)+1
1730      Budget$(I,1+Index(I))=Input$
1740      PRINT TAB(5);Subindex$[Index(I),Index(I)];".  ";Budget$(I,1+
Index(I))
1750      Input$=Flag$
1760      GOTO Add


************************


1780 Delete: PRINT PAGE;"The main category and each of its subcategori
es appear"
1790         PRINT "below.  To delete a subcategory, enter its letter"
1800         PRINT "(i.e. a or b, for example) and press CONT.",LIN(2)
1810         GOSUB Categoryprint
1820         INPUT "Which subcategory do you wish to delete?",C$
1830         Pos=POS(Subindex$,C$)
1840         IF Pos=0 THEN Pos=POS(Subindexc$,C$)
1850         IF (Pos<>0) AND (Pos<=Index(I)) THEN Delete1
1860         PRINT "There is no subcategory ";C$
1870         DISP "There is no subcategory ";C$
1880         WAIT 1000
1890         GOTO 1580
```

```
************************


1910 Delete1: FOR L=Pos+2 TO Index(I)+1
1920          Budget$(I,L-1)=Budget$(I,L)
1930          NEXT L
1940          Index(I)=Index(I)-1
1950          PRINT "Category ";C$;" has been deleted."
1960          PRINT "New category listing:",LIN(1)
1970          GOSUB Categoryprint
1980          GOTO 1580


************************


2000 Endloop2:  NEXT I
2010            GOTO 1280


************************


2030 Categoryprint: PRINT Budget$(I,1)
2040                FOR J=1 TO Index(I)
2050                PRINT TAB(5);Subindex$[J,J];".   ";Budget$(I,J+1)
2060                NEXT J
2070                RETURN


************************


2090 Nochange: PRINT PAGE;"Please wait a few moments for your data to
be stored"
2100           PRINT "on the particular data file and mass storage dev
ice you specified"
```

```
2110            PRINT "earlier.  You will be informed when the process
is complete."
2120            DISP "Your data is now being stored on file ";File$;"."
2130            READ #1,1
2140            MAT PRINT #1;Budget$,Index
2150            DISP ""
2160            IF Flagflag THEN 2190
2170            Flagflag=1
2180            RETURN
2190            GOSUB Prompt
2200 RETURN


************************


2220 Formprint: PRINTER IS Select
2230             FOR I=1 TO 14
2240                PRINT I;".  ";Budget$(I,1)
2250                FOR J=1 TO Index(I)
2260                   PRINT TAB(5);Subindex$[J,J];".  ";Budget$(I,J+1)
2270                NEXT J
2280                PRINT LIN(2)
2290           NEXT I
2300           PRINTER IS 16
2310           RETURN


************************


2330 Readchr: READ #1,1
2340          MAT READ #1;Budget$
2350          MAT READ #1;Index
2360          RETURN
2370 Readnum: READ #1,14
2380          MAT READ #1;Budget
2390          RETURN
```

```
************************


2410 Selectmonth: PRINT PAGE;"Please enter the month for which you wis
h to "
2420 PRINT "establish a budget.  Respond with the number corresponding
 with"
2430 PRINT "that month (i.e., if you want January, enter 1; if you wan
t"
2440 PRINT "February, enter 2; August is 8; December is 12, etc.) and
press"
2450 PRINT "CONT."
2460 Mo=14
2470 INPUT "Please enter the number of the month you are using",Mo
2480 Mo=INT(Mo)
2490 IF (Mo<1) OR (Mo>12) THEN 2460
2500 RETURN


************************


2520 Forecast: Mode1=1
2530           Insert$="wish to spend on"
2540           GOTO Enterfigures
2550 Endmonth: Mode1=2
2560           Insert$="actually spent on"
2570 Enterfigures: GOSUB Selectmonth
2580      PRINT PAGE;"INSTRUCTIONS:"
2590      PRINT "As each category and subcategory are displayed, you m
ay"
2600      PRINT "enter the amount ";Insert$;" that particular"
2610      PRINT "item this month."
2620      FOR I=1 TO 14
2630      PRINT LIN(2),TAB(35),"PREDICTED          ACTUAL"
2640      PRINT Budget$(I,1)
2650         Budget(I,1,Mode1,Mo)=0
2660         FOR J=1 TO Index(I)
2670            DISP "CATEGORY:";Budget$(I,1);"----SUBCATEGORY:";Budge
t$(I,1+J);
2680            INPUT Budget(I,1+J,Mode1,Mo)
```

```
2690             Budget(I,1,Mode1,Mo)=Budget(I,1,Mode1,Mo)+Budget(I,1+J
,Mode1,Mo)
2700             PRINT USING 2710;Subindex$(J,JJ),Budget$(I,1+J)
2710             IMAGE #5X,A,".   ",20A
2720             PRINT USING 2730;Budget(I,1+J,1,Mo),Budget(I,1+J,2,Mo)
2730             IMAGE 5X,DDDD.DD,10X,DDDD.DD
2740          NEXT J
2750       NEXT I
2760       PRINTER IS Select
2770       PRINT "Budget for the Month of ";Months$(Mo),LIN(2)
2780       GOSUB Print
2790       IF Mode1=1 THEN GOSUB Estimate
2800       GOSUB Prompt
2810 RETURN


************************


2830 Editfore:  Mode1=1
2840            GOTO Go
2850 Editexpen: Mode1=2
2860 Go:   GOSUB Selectmonth
2870       FOR I=1 TO 14
2880       PRINT LIN(2)
2890       GOSUB Smallprint
2900       IF Index(I)=0 THEN Nexti
2910       A$="N"
2920       DISP "Do you want to edit anything under ";Budget$(I,1);" (Y
/N)";
2930       INPUT A$
2940       IF (A$="N") OR (A$="n") THEN Nexti
2950       IF (A$="Y") OR (A$="y") THEN Which
2960       GOTO 2910
2970 Which: Mode$="PREDICTED:"
2980       IF Mode1=2 THEN Mode$="ACTUAL:"
2990       Budget(I,1,Mode1,Mo)=0
3000       PRINT LIN(2),TAB(35),"PREDICTED          ACTUAL"
3010       PRINT Budget$(I,1)
3020       FOR J=1 TO Index(I)
3030          DISP Mode$;"       ";Budget$(I,1+J);"=";Budget(I,1+J,Mode1
,Mo);
3040          INPUT Budget(I,1+J,Mode1,Mo)
3050          Budget(I,1,Mode1,Mo)=Budget(I,1,Mode1,Mo)+Budget(I,1+J,Mo
de1,Mo)
```

```
3060          PRINT USING 3400;Subindex$[J,J],Budget$(I,1+J)
3070          PRINT USING 3420;Budget(I,1+J,1,Mo),Budget(I,1+J,2,Mo)
3080       NEXT J
3090       GOSUB Smallprint
3100 Nexti: NEXT I
3110       PRINTER IS Select
3120       PRINT "Budget for the Month of ";Months$(Mo),LIN(2)
3130       GOSUB Print
3140       IF Mode1=1 THEN GOSUB Estimate
3150       GOSUB Prompt
3160 RETURN


************************


3180 Estimate: PRINTER IS Select
3190           PRINT LIN(2);"For the month of ";Months$(Mo);":";LIN(1)
3200           Totalout=0
3210           FOR I=1 TO 13
3220           Totalout=Totalout+Budget(I,1,1,Mo)
3230           NEXT I
3240           PRINT USING 3260;"ESTIMATED INCOME:       ",Budget(14,1
,1,Mo)
3250           PRINT USING 3260;"ESTIMATED EXPENDITURES: ",Totalout
3260           IMAGE K,5D.DD
3270           PRINT USING 3280
3280           IMAGE 31("-")
3290           Totalout=Budget(14,1,1,Mo)-Totalout
3300           PRINT USING 3260;"AMOUNT LEFT OVER:       ",Totalout
3310           PRINT LIN(2)
3320           PRINTER IS 16
3330           RETURN


************************


3350 Smallprint: ! "I" should be set before branching here
3360 PRINT LIN(2),TAB(35),"PREDICTED          ACTUAL"
3370 PRINT Budget$(I,1)
3380 FOR J=1 TO Index(I)
3390    PRINT USING 3400;Subindex$[J,J],Budget$(I,1+J)
```

```
3400     IMAGE #5X,A,".  ",20A
3410     PRINT USING 3420;Budget(I,1+J,1,Mo),Budget(I,1+J,2,Mo)
3420     IMAGE 5X,DDDD.DD,10X,DDDD.DD
3430 NEXT J
3440 RETURN


************************


3460 Print: !
3470 PRINTER IS Select
3480 PRINT TAB(35),"PREDICTED          ACTUAL"
3490 FOR I=1 TO 14
3500     PRINT I;".    ";Budget$(I,1)
3510     FOR J=1 TO Index(I)
3520        PRINT USING 3400;Subindex$[J,J],Budget$(I,1+J)
3530        PRINT USING 3420;Budget(I,1+J,1,Mo),Budget(I,1+J,2,Mo)
3540     NEXT J
3550     PRINT LIN(2)
3560 NEXT I
3570 PRINTER IS 16
3580 RETURN


************************


3600 Reconcile:  GOSUB Selectmonth
3610 PRINTER IS Select
3620 PRINT "For the month of ";Months$(Mo);":",LIN(1)
3630 PRINT TAB(35),"PREDICTED       ACTUAL          SURPLUS"
3640 Totalout=0
3650 FOR I=1 TO 14
3660     Incswitch=1
3670     IF I=14 THEN Incswitch=-1
3680     PRINT I;".  ";Budget$(I,1)
3690     FOR J=1 TO Index(I)
3700        PRINT USING 3710;Subindex$[J,J],Budget$(I,1+J)
3710        IMAGE #,5X,A,".  ",20A
3720        PRINT USING 3730;Budget(I,1+J,1,Mo),Budget(I,1+J,2,Mo),(Bud
get(I,1+J,1,Mo)-Budget(I,1+J,2,Mo))*Incswitch
3730        IMAGE 4X,M4D.DD,6X,M4D.DD,6X,M4D.DD
```

```
3740     NEXT J
3750     PRINT USING 3760
3760     IMAGE 30X,"----------------------------------------"
3770     PRINT USING 3780;Budget(I,1,1,Mo),Budget(I,1,2,Mo),(Budget(I,1
,1,Mo)-Budget(I,1,2,Mo))*Incswitch
3780     IMAGE "Totals:",20X,3(6X,M4D.DD),/,/,/,/
3790        IF I=14 THEN 3810
3800        Totalout=Totalout+Budget(I,1,2,Mo)
3810 NEXT I
3820 PRINT "For the month of ";Months$(Mo);":",LIN(1)
3830 PRINT USING 3850;Totalout
3840 PRINT USING 3860;Budget(14,1,2,Mo)
3850 IMAGE "TOTAL EXPENDITURES:",DDDDD.DD
3860 IMAGE "TOTAL INCOME:      ",DDDDD.DD
3870 IMAGE "----------------------------",/,"AMOUNT LEFT OVER:  ",DDD
DD.DD
3880 PRINT USING 3870;Budget(14,1,2,Mo)-Totalout
3890 PRINT LIN(2)
3900 PRINTER IS 16
3910 GOSUB Prompt
3920 RETURN


************************


3940 Review: GOSUB Selectmonth
3950         PRINTER IS Select
3960         PRINT "REVIEW OF BUDGET FOR THE MONTH OF ";Months$(Mo)
3970         GOSUB Print
3980         PRINTER IS 16
3990         GOSUB Prompt
4000 RETURN


************************


4020 Year: PRINT LIN(2);"For the year:",LIN(1)
4030       Totalout=Totalin=0
4040       FOR J=1 TO 12
4050       FOR I=1 TO 14
4060       IF I<>14 THEN Totalout=Totalout+Budget(I,1,1,J)
```

```
4070        IF I=14 THEN Totalin=Totalin+Budget(I,1,1,J)
4080        NEXT I
4090        NEXT J
4100        PRINT USING 3260;"TOTAL INCOME:       ",Totalin
4110        PRINT USING 3260;"TOTAL EXPENDITURES:",Totalout
4120        PRINT USING 3280
4130        PRINT USING 3260;"AMOUNT LEFT OVER:   ",Totalin-Totalout
4140        PRINT LIN(2)
4150        GOSUB Prompt
4160        RETURN


************************


4180 Store: PRINT PAGE;"Please wait for the data to be stored"
4190        DISP "Please wait for the data to be stored"
4200        READ #1,1
4210        MAT PRINT #1;Budget$,Index
4220        READ #1,14
4230        MAT PRINT #1;Budget
4240        PRINT PAGE
4250        DISP "The data is stored and the program is finished."
4260 END
```

User Instructions:

1. Insert the Utility Routines cartridge 2 into the primary transport (i.e., the transport above the special function keys).

2. Load the file :
   a. Type : LOAD "BUDGET:T15"
   b. Press : EXECUTE
3. Start the program :
   a. Press : RUN
4. When "Printer select code?" appears in the display area :
   a. Enter : The select code of the printer (Note : The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press : CONT
5. When "Please enter the file name" appears in the display area :
   a. Enter : The file name of your data file (Note : If the file does not exist yet, the program will build it.)
   b. Press : CONT
6. When "Please enter the device code of your storage medium" appears in the display area :
   a. Enter : The device code of your storage medium (e.g., T15, F8, D10 , etc.)
   b. Press : CONT
7. When "Is *filename* okay (Y/N)?" appears in the display area :
   a. If *filename* is the correct file name and device code:
      1) Enter : Y
      2) Press : CONT
      3) Go to step 8.

      or
   a. If *filename* is incorrect:
      1) Enter : N
      2) Press : CONT
      3) Go to step 5.

8. a. If this is the first time you have used the file name you entered in step 5, go to step 9.
   b. If you have already used the file name before, go to step 19.
9. When the message "Please wait while your data file is being initialized. You will receive further instructions when everything is ready " is printed on the screen, wait until the prompt in step 10 is displayed, and proceed normally.
10. When "CATEGORY:____---- Please enter subcategory —" appears in the display area:
    a. If you want (another) subcategory under the given category:
       1) Type: The name of the (next) subcategory. (Note: Subcategory names may be up to 20 characters long).
       2) Press: CONT
       3) Repeat step 10 as often as necessary.

       or

    a. If you do not want (another) subcategory under the given category:
       1) Press: CONT
       2) Repeat step 10 as often as necessary.
11. When "Do you wish to make any changes in the budget categories (Y/N)?" appears in the display area:
    a. If you wish to make any changes in the budget categories:
       1) Enter: Y
       2) Press: CONT
       3) Go to step 12.

       or

    a. If you do not wish to make any changes in the budget categories:
       1) Enter: N
       2) Press: CONT
       3) Go to step 18.

12. A category and its subcategories will be printed on the CRT. When "Do you wish to make any changes under the category ____ (Y/N)?" appears in the display area:
    a. If you wish to make any changes under the given category:
        1) Enter: Y
        2) Press: CONT
        3) Go to step 13.
        or
    a. If you do not wish to make any changes under the given category:
        1) Enter: N
        2) Press: CONT
        3) Repeat step 12 as often as necessary. After the last category has been entered, go to step 11.
13. Each subcategory will be displayed in the input area with the prompt "Make any changes you want and press CONT" in the display area:
    a. Make any changes you want in the subcategory.
    b. Press: CONT
    c. Repeat step 13 as often as necessary.
14. When "Do you wish to make any additions or deletions (Y/N)?" appears in the display area of the CRT:
    a. If you want to make any additions or deletions to the category:
        1) Enter: Y
        2) Press: CONT
        3) Go to step 15.
        or
    a. If you do not want to make any additions or deletions to the category:
        1) Enter: N
        2) Press: CONT
        3) Go to step 12 (unless the last category was under consideration, then go to step 11.
15. When "Do you wish to add or delete (A/D)?" appears in the display area:

a. If you wish to add subcategories:
   1) Enter: A
   2) Press: CONT
   3) Go to step 16.

b. If you wish to delete subcategories:
   1) Enter: D
   2) Press: CONT
   3) Go to step 17.

16. When "CATEGORY:_____-----Please enter subcategory__" appears in the display area:
   a. If you wish to add (another) subcategory:
      1) Type: The name of the subcategory.
      2) Press: CONT
      3) Repeat this step as often as you want (Note: You may not have more than a total of ten subcategories.)
      or
   a. If you do not wish to add more subcategories:
      1) Press: CONT
      2) Go to step 14.

17. The main category and each of its subcategories will be printed on the screen. Each of the subcategories will have a letter (A through J) corresponding to it. When "Which subcategory do you wish to delete?" appears in the display area:
   a. Enter: The letter corresponding to the subcategory you wish to delete (Note: If you change your mind and decide that you do not want to delete anything after all, enter a letter that does not have a corresponding subcategory.)
   b. Press: CONT
   c. Go to step 14.

18. When "Your data is now being stored on file ____" appears in the display area:
   a. Wait until the data is stored and the next prompt appears in the display area.
   b. Proceed to step 20.

19. When "Please wait until the data has been read in" appears in the display area:
    a. Wait until the data is read in and the next prompt appears in the display area.
    b. Proceed to step 20.
20. When "Select a key" appears in the display area:
    a. Press: One of the special function keys.
        1) If you want to initialize your budget subcategories:
            a) Press: Special Function key Ø.
            b) Go to step 10.
        2) If you want to edit the budget subcategories:
            a) Press: Special Function key 8.
            b) Go to step 11.
        3) If you want to enter a monthly forecast:
            a) Press: Special Function key 1.
            b) Go to step 21.
        4) If you want to edit a monthly forecast:
            a) Press: Special Function key 9.
            b) Go to step 29.
        5) If you want to enter your monthly expenditures:
            a) Press: Special Function key 2.
            b) Go to step 23.
        6) If you want to edit your monthly expenditures:
            a) Press: Special Function key 10.
            b) Go to step 32.
        7) If you want to reconcile a monthly budget with the corresponding expenditures:
            a) Press: Special Function key 3.
            b) Go to step 25.
        8) If you want a printed report of a past budget:
            a) Press: Special Function key 4.
            b) Go to step 27.
        9) If you want to add year-to-date expenditures:
            a) Press: Special Function key 11.
            b) Go to step 35.
        10) To end the program:
            a) Press: Special Function key 15.
            b) Go to step 36.

21. When "Please enter the number of the month you are using" appears in the display area :
    a. Enter : The number of the month you are using (i.e., 8 for August, 2 for February, etc.).
    b. Press : CONT
22. For each category and subcategory, the prompt "CATEGORY: _____--- SUBCATEGORY:_____?" appears in the display area:
    a. Enter: The amount of money you think will be spent for the given subcategory.
    b. Press: CONT
    c. Repeat step 22 until each category and subcategory have been gone through. Then go back to step 20.
23. When "Please enter the number of the month you are using" appears in the display area:
    a. Enter: The number of the month you are using (i.e., 8 for August, 2 for February, etc.).
    b. Press: CONT
24. For each category and subcategory, the prompt "CATEGORY: _____---SUBCATEGORY :_____?" will appear in the display area :
    a. Enter : The amount of money that was spent on the given subcategory.
    b. Press : CONT
    c. Repeat step 24 until each category and subcategory have been gone through. Then go back to step 20.
25. When "Please enter the number of the month you are using" appears in the display area :
    a. Enter : The number of the month you are using (i.e., 8 for August, 2 for February, etc.).
    b. Press : CONT
26. Wait until the report is printed. Then go back to step 20.
27. When "Please enter the number of the month you are using" appears in the display area :
    a. Enter : The number of the month you are using (i.e., 8 for August, 2 for February, etc.).
    b. Press : CONT

28. After the review is printed, go back to step 20.

29. When "Please enter the number of the month you are using" appears in the display area :

    a. Enter : The number of the month you are using (i.e., 8 for August, 2 for February, etc.).

    b. Press : CONT

30. When "Do you want to edit anything under___(Y/N)?" appears in the display area :

    a. If you want to edit any of the forecast amounts under the given category :

        1) Enter : Y

        2) Press : CONT

        3) Go to step 31

        or

    a. If you do not want to edit any of the forecast amounts under the given category :

        1) Enter : N

        2) Press : CONT

        3) Repeat step 30 until each of the categories has been gone through , then go to step 20.

31. When "PREDICTED *subcategory* = *amount?*" appears in the display area:

    a. If you wish to change the predicted amount:

        1) Enter: The correct amount.

        2) Press: CONT

        3) Repeat step 31 for each subcategory under the given category. Then go back to step 30 (unless the last category has been gone through, then go back to step 20).

        or

    a. If you do not wish to change the forecast amount:

        1) Press: CONT

        2) Repeat step 31 for each subcategory under the given category. Then go back to step 30 (unless the last category has been gone through, then go back to step 20).

32. When "Please enter the number of the month you are using" appears in the display area:
    a. Enter: The number of the month you are using (i.e., 8 for August, 2 for February, etc.).
    b. Press: CONT
33. When "Do you want to edit anything under____(Y/N)?" appears in the display area:
    a. If you want to edit any of the expenditures under the given category:
        1) Enter: Y
        2) Press: CONT
        3) Go to step 34.

        or

    a. If you do not want to edit any of the expenditures under the given category:
        1) Enter: N
        2) Press: CONT
        3) Repeat step 33 until each of the categories has been gone through, then go to step 20.
34. When "ACTUAL: *subcategory = amount?*" appears in the display area :
    a. If you wish to change the actual amount spent in the given subcategory :
        1) Enter : The correct amount.
        2) Press : CONT
        3) Repeat step 34 for each subcategory under the given category. Then go back to step 33 (unless the last category has been gone through, then go back to step 20).

        or

    a. If you do not wish to change the actual amount spent in the given subcategory :
        1) Press : CONT
        2) Repeat step 34 for each subcategory under the given category. Then go back to step 33 (unless the last category has been gone through, then go back to step 20).

35. The year-to-date expenditures in your file will be subtracted from your year-to-date income and the difference will be printed. After this is done, go back to step 20.
36. When "Please wait for the data to be stored" appears in the display area, do not remove your mass storage medium from its drive until you are told the program is finished.

## EXAMPLE

For the month of December:

| | PREDICTED | ACTUAL | SURPLUS |
|---|---|---|---|
| 1 . Shelter | | | |
| a. Rent | 200.00 | 200.00 | 0.00 |
| b. Electricity | 25.00 | 23.18 | 1.82 |
| c. Phone | 20.00 | 22.37 | -2.37 |
| Totals: | 245.00 | 245.55 | -.55 |
| 2 . Food and Groceries | | | |
| a. Food | 170.00 | 160.84 | 9.16 |
| b. Cosmetics | 12.00 | 5.78 | 6.22 |
| c. Household Goods | 15.00 | 10.59 | 4.41 |
| Totals: | 197.00 | 177.21 | 19.79 |
| 3 . Automobile Expenses | | | |
| a. Gas and oil | 30.00 | 38.54 | -8.54 |
| b. Tires | 0.00 | 0.00 | 0.00 |
| c. Repairs | 20.00 | 25.98 | -5.98 |
| Totals: | 50.00 | 64.52 | -14.52 |
| 4 . Clothing | | | |
| a. Husband | 50.00 | 22.15 | 27.85 |
| b. Wife | 50.00 | 48.67 | 1.33 |
| c. Children | 50.00 | 67.84 | -17.84 |
| Totals: | 150.00 | 138.66 | 11.34 |
| 5 . Entertainment | | | |
| a. Movies | 10.00 | 12.00 | -2.00 |
| b. Dinners | 20.00 | 26.84 | -6.84 |
| c. Skiing | 100.00 | 115.67 | -15.67 |
| d. Tennis | 0.00 | 0.00 | 0.00 |
| e. Travel | 75.00 | 46.88 | 28.12 |
| Totals: | 205.00 | 201.39 | 3.61 |

| | | | |
|---|---|---|---|
| 6 . Vices | | | |
| a. Liquor | 10.00 | 18.66 | -8.66 |
| b. Tobacco | 5.00 | 4.27 | .73 |
| c. Poker Money | 5.00 | -9.58 | 14.58 |
| Totals: | 20.00 | 13.35 | 6.65 |
| 7 . Contributions | | | |
| a. Church | 20.00 | 20.00 | 0.00 |
| b. Charity | 20.00 | 20.00 | 0.00 |
| Totals: | 40.00 | 40.00 | 0.00 |
| 8 . Insurance Premiums | | | |
| a. Life | 20.00 | 25.00 | -5.00 |
| b. Auto | 20.00 | 20.00 | 0.00 |
| c. Theft | 10.00 | 15.00 | -5.00 |
| d. Health | 20.00 | 20.00 | 0.00 |
| Totals: | 70.00 | 80.00 | -10.00 |
| 9 . Medical Bills | | | |
| a. Dentist | 100.00 | 115.25 | -15.25 |
| b. Gynecologist | 30.00 | 25.00 | 5.00 |
| c. Orthodontist | 50.00 | 68.79 | -18.79 |
| Totals: | 180.00 | 209.04 | -29.04 |
| 10 . Gifts | | | |
| a. Birthdays | 20.00 | 12.00 | 8.00 |
| b. Weddings | 10.00 | 13.54 | -3.54 |
| c. Christmas | 200.00 | 297.82 | -97.82 |
| d. Anniversaries | 0.00 | 0.00 | 0.00 |
| Totals: | 230.00 | 323.36 | -93.36 |

| | | | |
|---|---|---|---|
| 11 . Future Investments | | | |
| a. Savings | 100.00 | 50.00 | 50.00 |
| b. Stock | 100.00 | 0.00 | 100.00 |
| Totals: | 200.00 | 50.00 | 150.00 |
| 12 . Installment Payments | | | |
| a. Furniture | 25.00 | 25.00 | 0.00 |
| b. Car | 75.00 | 75.00 | 0.00 |
| Totals: | 100.00 | 100.00 | 0.00 |
| 13 . Miscellaneous | | | |
| a. Lunch Money | 25.00 | 30.00 | -5.00 |
| b. Fudge factor | 30.00 | 0.00 | 30.00 |
| c. Taxes | 0.00 | 0.00 | 0.00 |
| Totals: | 55.00 | 30.00 | 25.00 |
| 14 . Income | | | |
| a. Husband's salary | 1000.00 | 998.47 | -1.53 |
| b. Wife's salary | 700.00 | 705.64 | 5.64 |
| c. Interest | 25.00 | 32.45 | 7.45 |
| d. Tax refunds | 0.00 | 0.00 | 0.00 |
| Totals: | 1725.00 | 1736.56 | 11.56 |

For the month of December:

TOTAL EXPENDITURES: 1673.08
TOTAL INCOME: 1736.56
---
AMOUNT LEFT OVER: 63.48

Subprogram Name : Meanvariance

This subprogram will take a two-dimensional matrix and return two vectors; one vector will contain the means of the rows (or columns) of the matrix, and the other will contain the variances of the rows (or columns) of the matrix.

Subprogram Utilization :

File Name : MATSTT

Calling Syntax : CALL Meanvariance (Array(*), Mean(*), Variance(*), Rowcol, Lowrow, Highrow, Lowcol, Highcol)

Input Parameters:

Array(*) - Array(*) is a full-precision two-dimensional array. It must be allocated in the user's calling program.

Rowcol - This is a full-precision number telling the subprogram whether to find the means and variances of the rows of Array(*) or the columns of Array(*). If Rowcol is 0, then the means and variances of the columns will be returned. If Rowcol is 1, then the means and variances of the rows will be returned. If Rowcol is anything else, nothing at all will be done.

Lowrow - This full precision number is the lower row subscript of Array(*).

Highrow - This full precision number is the upper row subscript of Array(*).

Lowcol - This full precision number is the lower column subscript of Array(*).

Highcol - This full precision number is the upper column subscript of Array(*).

Output Parameters:

Mean(*) and Variance(*) - Mean(*) and Variance (*) are full-precision one-dimensional arrays. They must be allocated in the user's calling program. Furthermore, if Rowcol is 0 (see the section under Input Parameters), then the subscript ranges of these two arrays must contain Lowcol and Highcol (again, refer to the section under Input Parameters). On the other hand, if Rowcol is 1, then the subscript ranges of these arrays must contain Lowrow and Highrow. After execution of this subprogram, Mean(i) will contain the mean of the ith row (or column) of Array(*) and Variance (i) will contain the variance of the ith row (or column) of Array(*).

Local Variables :

Array(*) - See the section under Input Parameters

High - The upper subscript of Mean(*) and Variance (*)
If Rowcol is 0 then High = Highcol
If Rowcol is 1 then High = Highrow

Highcol - See the section under Input Parameters

Highn - If Rowcol is 0 then Highn = Highrow
If Rowcol is 1 then Highn = Highcol

Highrow - See the section under Input Parameters

I - Loop counter

J - Loop counter

Low - The lower subscript of Mean(*) and Variance (*)
If Rowcol is 0 then Low = Lowcol
If Rowcol is 1 then Low = Lowrow

| | | |
|---|---|---|
| Lowcol | - | See the section under Input Parameters |
| Lown | - | If Rowcol is 0 then Lown = Lowrow<br>If Rowcol is 1 then Lown = Lowcol |
| Lowrow | - | See the section under Input Parameters |
| Mean(*) | - | See the section under Output Parameters |
| N | - | If Rowcol is 0 then N = Highrow + 1 - Lowrow<br>If Rowcol is 1 then N = Highcol + 1 - Lowcol |
| Rowcol | - | See the section under Input Parameters |
| Variance(*) | - | See the section under Output Parameters |
| X1 | - | Temporary storage for elements of Array(*) |
| X2 | - | $\Sigma x$ |
| X | | $\Sigma x^2$ |

Special Considerations and Programming Hints :

1. Be sure to read the sections under Input and Output Parameters carefully. It is important that you understand the dimensioning of the arrays Mean(*) and Variance (*).
2. System Configuration:
   Standard Memory Option
   (Optional) Printer

## Annotated Listing of Subprogram Meanvariance

```
10   SUB Meanvariance(Array(*),Mean(*),Variance(*),Rowcol,Lowrow,Highr
ow,Lowcol,Highcol)
20   IF (Rowcol<>0) AND (Rowcol<>1) THEN SUBEXIT
30   !  Rowcol=0 -> Return means and variances for each column in Arra
y(*)
40   !  Rowcol=1 -> Return means and variances for each row in Array(*
)
60   IF Rowcol THEN Setrow
70 Setcol:   Low=Lowcol       ! Determine bounds if by columns
80           High=Highcol
90           Lown=Lowrow
100          Highn=Highrow
110          GOTO Compute
120 Setrow:  Low=Lowrow       ! Determine bounds if by rows
130          High=Highrow
140          Lown=Lowcol
150          Highn=Highcol
160 Compute: N=Highn+1-Lown   ! Compute the number of elements that go
into
161          IF N>1 THEN 170 !  the computation of each mean and varia
nce
162          BEEP             ! Check for illegal variance computation
163          DISP "THERE MUST BE AT LEAST TWO ELEMENTS FOR VARIANCE FO
RMULA.  SUBPROGRAM ENDED."
164          WAIT 1000
165          SUBEXIT
170          FOR J=Low TO High
180             X1=X2=0
190             FOR I=Lown TO Highn
200                X=Array(Rowcol*J+NOT Rowcol*I,Rowcol*I+NOT Rowcol*J
)
210                X1=X1+X     ! Sum of X's
220                X2=X2+X*X   ! Sum of X squares
230             NEXT I
240             Mean(J)=X1/N   ! Compute mean
250             Variance(J)=(X2-X1*X1/N)/(N-1)    ! Compute variance
260          NEXT J
270          SUBEXIT
```

Methods and Formulae :

mean : $$\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j$$

variance: $$s^2 = \frac{\sum_{j=1}^{n} x_j^2 - \frac{1}{n} \left( \sum_{j=1}^{n} x_j \right)^2}{(n-1)}$$

REFERENCES:

1. Guttman, Wilks, Hunter, Introductory Engineering Statistics (John Wiley and Sons, Inc., 1971) pp. 69,71.

Driver Utilization:

File Name: MATDRI

This program allows the user to enter any two-dimensional array (or have the machine generate a random one). The program will call the subprogram Meanvariance which will generate the means and variances of the rows (or columns) of the array, depending upon what the user desires.

User Instructions:

1. Insert the Utility Routines cartridge 2 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "MATDRI:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
   c. If you entered 16 in part a, go to step 5.
   d. If you did not enter 16 in part a, then when "Do you want top-of-form generation on your output (Y/N)?" appears in the display area:
      1) If you want page feeds in your output:
         a) Enter: Y
         b) Press: CONT
         c) Go to step 5.

         or
      2) If you do not want page feeds:
         a) Enter: N
         b) Press: CONT
         c) Go to step 5.

5. When "Enter the lower row subscript" appears in the display area :
   a. Enter : The lower row boundary of the array.
   b. Press : CONT
6. When "Enter the upper row subscript" appears in the display area :
   a. Enter : The upper row boundary of the array.
   b. Press : CONT
7. When "Enter the lower column subscript" appears in the display area :
   a. Enter : The lower column boundary of the array.
   b. Press : CONT
8. When "Enter the upper column subscript" appears in the display area :
   a. Enter : The upper column boundary of the array.
   b. Press : CONT
9. When "Enter your own matrix (Y/N)?" appears in the display area :
   a. If you wish to enter your own matrix :
      1) Enter : Y
      2) Press : CONT
      3) Go to step 10.

      or

   a. If you do not wish to enter your own matrix :
      1) Enter : N
      2) Press : CONT
      3) Go to step 12.
10. When "ELEMENT # i, j" appears in the display area :
   a. Enter : The matrix element belonging in the jth column of the ith row.
   b. Press : CONT
   c. Repeat step 10 as often as necessary.
11. When "Enter the subscripts" appears in the display area :
   a. If you wish to make any changes in the matrix :
      1) Enter : The subscripts of the element you wish to change separated by a comma (e.g.,1,1).

2) Press : CONT

3) Go to step 10.

or

a. If you do not wish to make any more changes in the matrix :

1) Press : CONT

2) Go to step 12

12. When "Means and variances on row or column ( R/C)?" appears in the display area :

a. If you want the program to find the means and variances of the rows of the array :

1) Enter : R

2) Press : CONT

or

a. If you want the program to find the means and variances of the columns of the array :

1) Enter : C

2) Press : CONT

13. When "Do you want to run the program again (Y/N)?" appears in the display area :

a. If you want to run the program again :

1) Enter : Y

2) Press : CONT

3) Go to step 14.

or

a. If you do not want to run the program again :

1) Enter : N

2) Press : CONT

3) The program stops.

14. When "Do you want to use the old array (Y/N)?" appears in the display :

a. If you want to use the old array (or most of it—you will have the opportunity to edit the old array) :

1) Enter : Y

2) Press : CONT

3) Go to step 11.

or

a. If you want a completely new array (with the old dimensions)
   1) Enter : N
   2) Press : CONT
   3) Go to step 9.

ARRAY:

| | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 1 | 6.78E+02 | 3.82E+02 | 4.19E+02 | 5.03E+02 | 7.43E+02 | 5.09E+02 |
| 2 | 3.94E+02 | 3.87E+02 | 8.89E+02 | 8.10E+02 | 8.75E+02 | 9.79E+02 |
| 3 | 4.04E+02 | 3.15E+02 | 3.80E+01 | 6.97E+02 | 4.14E+02 | 8.62E+02 |
| 4 | 9.90E+02 | 7.30E+01 | 7.82E+02 | 8.93E+02 | 2.00E+02 | 2.73E+02 |
| 5 | 7.33E+02 | 9.13E+02 | 8.10E+01 | 9.34E+02 | 6.94E+02 | 8.52E+02 |
| 6 | 9.29E+02 | 4.27E+02 | 7.33E+02 | 8.38E+02 | 3.42E+02 | 7.30E+01 |
| 7 | 8.28E+02 | 6.30E+02 | 2.56E+02 | 1.10E+01 | 5.90E+02 | 2.84E+02 |
| 8 | 1.09E+02 | 9.17E+02 | 9.96E+02 | 6.54E+02 | 1.21E+02 | 9.25E+02 |
| 9 | 3.78E+02 | 9.39E+02 | 7.50E+01 | 5.63E+02 | 8.16E+02 | 3.14E+02 |
| 10 | 3.97E+02 | 1.09E+02 | 9.12E+02 | 2.35E+02 | 4.10E+02 | 2.31E+02 |
| 11 | 4.35E+02 | 9.50E+01 | 9.38E+02 | 1.56E+02 | 8.90E+02 | 7.29E+02 |
| 12 | 5.67E+02 | 4.53E+02 | 7.89E+02 | 5.00E+01 | 8.23E+02 | 7.13E+02 |
| 13 | 6.20E+01 | 1.80E+02 | 5.20E+01 | 9.03E+02 | 4.31E+02 | 9.80E+02 |
| 14 | 9.85E+02 | 7.82E+02 | 1.14E+02 | 6.65E+02 | 5.79E+02 | 6.50E+02 |
| 15 | 1.79E+02 | 8.38E+02 | 4.19E+02 | 6.60E+01 | 1.17E+02 | 8.65E+02 |
| 16 | 3.60E+01 | 4.48E+02 | 4.52E+02 | 4.86E+02 | 9.05E+02 | 7.34E+02 |
| 17 | 4.35E+02 | 1.90E+02 | 2.80E+02 | 5.71E+02 | 7.37E+02 | 9.88E+02 |
| 18 | 4.37E+02 | 8.38E+02 | 1.07E+02 | 7.60E+01 | 2.04E+02 | 8.30E+02 |
| 19 | 2.41E+02 | 7.31E+02 | 1.22E+02 | 8.87E+02 | 5.02E+02 | 9.63E+02 |
| 20 | 4.04E+02 | 8.79E+02 | 6.33E+02 | 5.50E+02 | 1.18E+02 | 2.80E+02 |

Means and variances of the columns of the matrix:

| | Means | Variances |
|---|---|---|
| 5 | 4.936E+02 | 8.83002E+04 |
| 6 | 5.263E+02 | 9.54818E+04 |
| 7 | 4.544E+02 | 1.20647E+05 |
| 8 | 5.274E+02 | 1.02636E+05 |
| 9 | 5.256E+02 | 7.75878E+04 |
| 10 | 6.517E+02 | 9.16217E+04 |

ARRAY:

| | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| 1 | 6.78E+02 | 3.82E+02 | 4.19E+02 | 5.03E+02 | 7.43E+02 | 5.09E+02 |
| 2 | 5.94E+02 | 3.87E+02 | 8.89E+02 | 8.10E+02 | 8.75E+02 | 9.79E+02 |
| 3 | 4.04E+02 | 3.15E+02 | 3.80E+01 | 6.97E+02 | 4.14E+02 | 8.62E+02 |
| 4 | 9.90E+02 | 7.30E+01 | 7.82E+02 | 8.93E+02 | 2.00E+02 | 2.73E+02 |
| 5 | 7.83E+02 | 9.13E+02 | 8.10E+01 | 9.34E+02 | 6.94E+02 | 8.52E+02 |
| 6 | 9.29E+02 | 4.27E+02 | 7.33E+02 | 8.38E+02 | 3.42E+02 | 7.30E+01 |
| 7 | 8.28E+02 | 6.30E+02 | 2.56E+02 | 1.10E+01 | 5.90E+02 | 2.84E+02 |
| 8 | 1.09E+02 | 9.17E+02 | 9.96E+02 | 6.54E+02 | 1.21E+02 | 9.25E+02 |
| 9 | 3.78E+02 | 9.39E+02 | 7.50E+01 | 5.63E+02 | 8.16E+02 | 3.14E+02 |
| 10 | 3.97E+02 | 1.09E+02 | 9.12E+02 | 2.35E+02 | 4.10E+02 | 2.31E+02 |
| 11 | 4.35E+02 | 9.50E+01 | 9.38E+02 | 1.56E+02 | 8.90E+02 | 7.29E+02 |
| 12 | 5.67E+02 | 4.53E+02 | 7.89E+02 | 5.00E+01 | 8.23E+02 | 7.13E+02 |
| 13 | 6.20E+01 | 1.80E+02 | 5.20E+01 | 9.03E+02 | 4.31E+02 | 9.80E+02 |
| 14 | 9.85E+02 | 7.82E+02 | 1.14E+02 | 6.65E+02 | 5.79E+02 | 6.50E+02 |
| 15 | 1.79E+02 | 8.38E+02 | 4.19E+02 | 6.60E+01 | 1.17E+02 | 8.65E+02 |
| 16 | 3.60E+01 | 4.48E+02 | 4.52E+02 | 4.86E+02 | 9.05E+02 | 7.34E+02 |
| 17 | 4.35E+02 | 1.90E+02 | 2.80E+02 | 5.71E+02 | 7.37E+02 | 9.88E+02 |
| 18 | 4.37E+02 | 8.38E+02 | 1.07E+02 | 7.60E+01 | 2.04E+02 | 8.30E+02 |
| 19 | 2.41E+02 | 7.31E+02 | 1.22E+02 | 8.87E+02 | 5.02E+02 | 9.63E+02 |
| 20 | 4.04E+02 | 8.79E+02 | 6.33E+02 | 5.50E+02 | 1.18E+02 | 2.80E+02 |

Means and variances of the rows of the matrix:

| | Means | Variances |
|---|---|---|
| 1 | 5.390E+02 | 2.04364E+04 |
| 2 | 7.557E+02 | 4.93799E+04 |
| 3 | 4.550E+02 | 8.43968E+04 |
| 4 | 5.352E+02 | 1.58102E+05 |
| 5 | 7.095E+02 | 1.02555E+05 |
| 6 | 5.570E+02 | 1.09140E+05 |
| 7 | 4.332E+02 | 9.02194E+04 |
| 8 | 6.203E+02 | 1.66777E+05 |
| 9 | 5.142E+02 | 1.05089E+05 |
| 10 | 3.823E+02 | 8.01695E+04 |
| 11 | 5.405E+02 | 1.34626E+05 |
| 12 | 5.658E+02 | 8.32826E+04 |
| 13 | 4.347E+02 | 1.73381E+05 |
| 14 | 6.292E+02 | 8.39214E+04 |
| 15 | 4.140E+02 | 1.29548E+05 |
| 16 | 5.102E+02 | 8.77322E+04 |
| 17 | 5.335E+02 | 8.82691E+04 |
| 18 | 4.153E+02 | 1.21189E+05 |
| 19 | 5.743E+02 | 1.18863E+05 |
| 20 | 4.773E+02 | 7.28575E+04 |

# Driver Listing

```
10    LINK "MATSTT",3000
20    DIM Top$[1]
30    PRINTER IS 16
40    PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is"
50    PRINT "running will appear on the CRT only.  The output for the p
rogram"
60    PRINT "will be printed on the device having the select code you a
re"
70    PRINT "asked to enter below.  If you press CONT without entering"
80    PRINT "a number, the output will appear on the default printer, t
he CRT"
90    PRINT "(select code 16).",LIN(4)
100   S=16
110   INPUT "Printer select code?",S
120   S=INT(S)
130   IF (S<0) OR (S>16) THEN 100
140   IF S<>16 THEN 200
150   PRINTER IS S       ! Check to be sure the printer is there
160   PRINT
170   PRINTER IS 16
180   Top$="N"
190   GOTO 280
200   PRINT PAGE;"Please indicate below whether or not the printer you
are using"
210   PRINT "has top-of-form generation and whether or not you wish to
make"
220   PRINT "use of that capability.  If you wish to make use of the to
p-of-form"
230   PRINT "generation, enter a Y and press CONT.  If you do not want
top-of-form"
240   PRINT "generation, enter an N and press CONT."
250   Top$="Y"
260   INPUT "Do you want top-of-form generation on your output (Y/N)?",
Top$
270   IF (Top$<>"y") AND (Top$<>"Y") AND (Top$<>"n") AND (Top$<>"N") TH
EN 250
280   PRINT PAGE
290   PRINT "Below you are asked to enter the lower and upper row and c
olumn"
300   PRINT "subscripts of the array you wish to analyze.  The subscrip
ts may be"
310   PRINT "positive, negative, or both.  The only restrictions are th
at the"
320   PRINT "lower row subscript must be less than or equal to the uppe
r row"
330   PRINT "subscript, and the lower column subscript must be less tha
```

```
n or"
340   PRINT "equal to the upper column subscript."
350   INPUT "Enter the lower row subscript",N1
360   INPUT "Enter the upper row subscript",N2
370   IF N1<=N2 THEN 420
380   BEEP
390   DISP "ILLEGAL RANGE"
400   WAIT 700
410   GOTO 350
420   INPUT "Enter the lower column subscript",M1
430   INPUT "Enter the upper column subscript",M2
440   IF M1<=M2 THEN 490
450   BEEP
460   DISP "INVALID RANGE"
470   WAIT 700
480   GOTO 420
490   CALL Rcdriver(N1,N2,M1,M2,S,Top$)
500   IF S=16 THEN 540
510   PRINTER IS S
520   IF UPC$(Top$)="N" THEN 540
530   PRINT PAGE
540   PRINTER IS 16
550 END
560 SUB Rcdriver(N1,N2,M1,M2,S,Top$)
570 DIM A(N1:N2,M1:M2),Meanrow(N1:N2),Meancol(M1:M2),Varrow(N1:N2),Var
col(M1:M2)
580       PRINT PAGE,"If you wish to enter your own array, enter Y in r
esponse"
590       PRINT "to the question below.  If you don't wish to enter you
r own"
600       PRINT "array, but would rather have the machine generate a ra
ndom array,"
610       PRINT "enter N.  If you press CONT without entering anything,
 Y"
620       PRINT "will be assumed as the response."
630       S$="Y"
640       INPUT "Enter your own matrix (Y/N)?",S$
650       IF (S$="Y") OR (S$="y") THEN Manual
660       IF (S$="N") OR (S$="n") THEN Auto
670       GOTO 630
680 Auto:  FOR I=N1 TO N2
690         FOR J=M1 TO M2
700         A(I,J)=INT(1000*RND)
710         NEXT J
720         NEXT I
730         GOTO Run
740 Manual: PRINT PAGE
```

```
750         PRINTER IS S
760         PRINT "       ";
770         FOR I=M1 TO M2
780         PRINT USING 790;I
790         IMAGE #,"    "MDD"    "
800         NEXT I
810         PRINT
820         FOR I=N1 TO N2
830         PRINT USING 840;I
840         IMAGE #," "MDD"   "
850         FOR J=M1 TO M2
860         DISP "ELEMENT #";I;J;
870         INPUT A(I,J)
880         PRINT USING 890;A(I,J)
890         IMAGE #,MD.2DE" "
900         NEXT J
910         PRINT
920         NEXT I
930      PRINTER IS 16
940      IF S<>16 THEN PRINT PAGE
950      PRINT LIN(1),"If you want to make any changes, enter the subs
cripts of the"
960      PRINT "element you wish to change.  Otherwise, press CONT."
970      PRINT "without entering anything.",LIN(2)
980      PRINTER IS S
990      I=J=3.141592654
1000     INPUT "Enter the subscripts",I,J
1010        IF (I=3.141592654) AND (J=3.141592654) THEN Run
1020        IF (I>=N1) AND (I<=N2) AND (J>=M1) AND (J<=M2) THEN 1070
1030        BEEP
1040        DISP "ILLEGAL SUBSCRIPTS"
1050        WAIT 700
1060        GOTO 990
1070        DISP "Element #("&VAL$(I)&","&VAL$(J)&")";
1080        INPUT A(I,J)
1090        PRINT "Element #("&VAL$(I)&","&VAL$(J)&") IS "&VAL$(A(I,J)
)
1100        GOTO 990
1110 Run: PRINTER IS S
1120     IF (S<>16) AND ((Top$="Y") OR (Top$="y")) THEN PRINT PAGE;
1130     PRINT "ARRAY:"
1140     CALL Output(A(*),N1,N2,M1,M2)
1150     PRINTER IS 16
1160     IF S<>16 THEN PRINT PAGE;
1170     IF S=16 THEN PRINT LIN(2);
1180     PRINT "     Below, you are asked to decide whether you want t
o find"
```

```
1190      PRINT "the means and variances of the rows of the array or th
e "
1200      PRINT "columns of the array.  Respond with R for rows or C fo
r columns."
1210      PRINT "If you press CONT without entering anything, R will be
 the assumed"
1220      PRINT "response.",LIN(2)
1230      S$="R"
1240      INPUT "Means and variances on row or column (R/C)?",S$
1250      IF (S$="R") OR (S$="r") THEN 1300
1260      IF (S$="C") OR (S$="c") THEN 1370
1270      S$="R"
1280      BEEP
1290      GOTO 1240
1300      CALL Meanvariance(A(*),Meanrow(*),Varrow(*),1,N1,N2,M1,M2)
1310      IF S<>16 THEN PRINT PAGE
1320      PRINTER IS S
1330      IF (Top$="y") OR (Top$="Y") THEN PRINT PAGE;
1340      PRINT LIN(1),"Means and variances of the rows of the matrix:"
,LIN(1)
1350      CALL Printresults(Meanrow(*),Varrow(*),N1,N2,1)
1360      GOTO 1430
1370      CALL Meanvariance(A(*),Meancol(*),Varcol(*),0,N1,N2,M1,M2)
1380      IF S<>16 THEN PRINT PAGE
1390      PRINTER IS S
1400      IF (Top$="y") OR (Top$="Y") THEN PRINT PAGE;
1410      PRINT LIN(1),"Means and variances of the columns of the matri
x:",LIN(1)
1420      CALL Printresults(Meancol(*),Varcol(*),M1,M2,0)
1430      PRINTER IS 16
1440      PRINT LIN(2),"   You may, at this point, run the program agai
n, using the"
1450      PRINT "same array, or you may change the array, as long as th
e dimensions"
1460      PRINT "remain the same.  If you want to run the program again
, enter Y"
1470      PRINT "and press CONT.  If you want to stop, enter N and pres
s CONT."
1480      PRINT "If you press CONT without entering anything, Y will be
 the assumed"
1490      PRINT "response."
1500      S$="Y"
1510      INPUT "Do you want to run the program again (Y/N)?",S$
1520      IF (S$="N") OR (S$="n") THEN SUBEXIT
1530      IF (S$="y") OR (S$="Y") THEN 1560
1540      BEEP
1550      GOTO 1500
```

```
1560      PRINT LIN(2),"   Now you are given the choice of using your o
ld array,  ( you"
1570      PRINT "may edit it), or you may generate a new array with the
 same dimensions."
1580      PRINT "If you wish to use the old array, enter Y and press CO
NT.  If you"
1590      PRINT "don't want to use the old array, but would rather gene
rate a com-"
1600      PRINT "pletely new array, enter N and press CONT.  If you pre
ss CONT without"
1610      PRINT "entering anything, Y will be the assumed response."
1620      S$="Y"
1630      INPUT "Do you want to use the old array (Y/N)?",S$
1640      IF (S$="N") OR (S$="n") THEN 580
1650      IF (S$="Y") OR (S$="y") THEN 1670
1660      GOTO 1620
1670      CALL Output(A(*),N1,N2,M1,M2)
1680      GOTO 950
1690 SUB Output(A(*),N1,N2,M1,M2)
1700       PRINT "         ";
1710       Linenumber=1
1720       FOR I=M1 TO M2
1730          IF (I+Linenumber-M1) MOD 8<>0 THEN 1760
1740          Linenumber=Linenumber+1
1750          PRINT LIN(1),"         ";
1760          PRINT USING 1770;I
1770          IMAGE #,3XM4D2X
1780       NEXT I
1790       PRINT LIN(1),"         ";
1800       FOR I=M1 TO M2
1810          PRINT USING 1820
1820          IMAGE #,10("-")
1830          IF I+1-M1>=7 THEN 1850
1840       NEXT I
1850       PRINT
1860       FOR I=N1 TO N2
1870          PRINT USING 1880;I
1880          IMAGE #,M4D" | "
1890          Linenumber=1
1900          FOR J=M1 TO M2
1910             IF (J+Linenumber-M1) MOD 8<>0 THEN 1940
1920             Linenumber=Linenumber+1
1930             PRINT LIN(1),"      | ";
1940             PRINT USING 1950;A(I,J)
1950             IMAGE #,MD.2DEX
1960          NEXT J
1970          PRINT
```

```
1980       NEXT I
1990 SUBEXIT
2000 SUB Printresults(M(*),V(*),Low,High,Rowcol)
2010 IF (Rowcol<>0) AND (Rowcol<>1) THEN SUBEXIT
2020       PRINT "                Means                          Variances"
2030       FOR I=Low TO High
2040          PRINT USING 2050;I,M(I),V(I)
2050          IMAGE DD,2X,"!",5X,MD.3DE,12X,MD.5DE
2060       NEXT I
2070       PRINT LIN(2)
2080       SUBEXIT
```

Subprograms : Linear
Logarithmic
Exponential
Polynomial

These subprograms take a set of (x,y) coordinates and fit various types of curves to them using least squares methods. In addition, an AOV (Analysis of variance) table is computed.

Subprogram Utilization:

File Name: REGS

(Note : All four subprograms are stored in the same file for ease of use. The Linear, Logarithmic, and Exponential subprograms are similar in their calling syntaxes and local variables used; hence they will be treated in the same section. The Polynomial subprogram will be dealt with separately).

Calling Syntax: CALL{Linear / Logarithmic / Exponential}(X(*), Y(*), N, A, B, Regss, Resss, Totalss, Regms, Resms, F, Dfreg, Dfres, Dftot, Abort)

Input Parameters:

X(*) - This full-precision one-dimensional array holds the X coordinates of the data points. It must have at least N elements.

Y(*) - This full-precision one-dimensional array holds the Y coordinates of the data points. It must have at least N elements.

N - This full-precision variable, constant, or expression tells how many data points are passed in through the X(*) and Y(*) arrays. N must be at least 3.

Output Parameters:

A - The coefficient A in the equation of the computed curve $\left\{\begin{array}{l} Y = A+B*X \\ Y = A+B*LOG(X) \\ Y = A*EXP(B*X). \end{array}\right\}$ A must be a full-precision variable.

B - The coefficient B in the equation of the computed curve $\left\{\begin{array}{l} Y = A+B*X \\ Y = A+B*LOG(X) \\ Y = A*EXP(B*X). \end{array}\right\}$ B must be a full-precision variable.

Regss - The entry in the AOV table for regression sum of squares. This argument must be a full-precision variable.

Resss - This full-precision variable gives the residual sum of squares.

Totalss - This full-precision variable gives the total sum of squares.

Regms - This full-precision variable gives the regression mean squares.

Resms - This full-precision variable gives the residual mean squares.

F - This full-precision variable gives the F statistic (Regms/Resms).

Dfreg - This full-precision variable gives the regression degrees of freedom (1 for these models).

Dfres - This full-precision variable gives the residual degrees of freedom (N-2).

Dftot - This full-precision variable gives the total degrees of Freedom (Dfreg + Dftot).

Abort - This full-precision variable is set to 1 if a fatal error occurs in the subprogram. (Probably from using negative data with the Logarithmic and Exponential Models). Abort is Ø if everything is normal.

Local Variables:

I - Used as a loop counter

Logx - (Only used in subprogram Logarithmic) LOG (Y(I))

Logy - (Only used in subprogram Exponential) LOG (Y(I))

X1 - Sum of Xs (or log(x(i)) in subprogram Logarithmic)

X2 - Sum of X squares

Y1 - Sum of Ys (or log(y(i)) in subprogram Exponential)

Y2 - Sum of Y squares

Z - Sum of XY products

(Note: The following section deals with subprogram Polynomial)

Calling Syntax: CALL Polynomial (X(*), Y(*), N, Degree, Coeff(*), Regss, Resss, Totalss, Regms, Resms, F, Dfreg, Dfres, Dftot, Abort)

Input Parameters:

X(*) - This full-precision one-dimensional array holds the X coordinates of the data points. It must have at least N elements.

Y(*) - This full-precision one-dimensional array holds the Y coordinates of the data points. It must have at least N elements.

N - This full-precision variable, constant, or expression tells the number of data points in the X(*) and Y(*) arrays. N must be at least 3.

Degree - This full-precision variable, constant, or expression tells the degree of the polynomial to be used. Degree may not exceeed N-2.

Output Parameters :

Coeffs(*) - This one-dimensional full-precision array holds the coefficients of the polynomial that has been fit to the data points. Coeffs(*) must have at least (Degree+1) elements. (The lower subscript must be at most ∅ and the upper subscript must be at least (Degrees)). The coefficient of the $x^i$ term of the polynomial is Coeffs(i).

Regss - The entry in the AOV table for regression sum of squares. This argument must be a full-precision variable.

Resss - This full-precision variable gives the residual sum of squares.

Totalss - This full-precision variable gives the total sum of squares.

Regms - This full-precision variable gives the regression mean squares.

Resms - This full-precision variable gives the residual mean squares.

F - This full-precision variable gives the F statistic (Regms/Resms).

Dfreg - This full-precision variable gives the regression degrees of freedom (Degree for polynomial model).

Dfres - This full-precision variable gives the residual degrees of Freedom (N-2).

Dftot - This full-precision variable gives the total degrees of Freedom (Dfreg + Dftot).

Abort - Abort is set to 1 if a fatal error occurs in the subprogram. Otherwise it is zero.

Local Variables:

B(*) - This full-precision one-dimensional array is used to hold the right-hand side of the system of equa-

tions that must be solved to yield the coefficients of the polynomial.

I - Loop counter

Inv(*) - This two-dimensional matrix is used to hold the inverse of the coefficient matrix of the system of equations.

J - Loop counter

K - Loop counter

L - Loop counter

M - Local variable of function FNG

Matrix(*) - The matrix holding the system of simultaneous equations whose solution is the array of coefficients of the polynomial.

X1 - Sum of X s

X2 - Sum of X squares

Y1 - Sum of Y s

Y2 - Sum of Y squares

Z - Sum of XY s

Special Considerations and Programming Hints :

1. The F ratio is an indication of how good the curve fit is. The higher the F ratio, the better the fit. The user would be wise to consult a table, as several factors enter into what is considered a good F ratio (e.g., the number of data points, the number of coefficients, etc).
2. The provided driver tests the Abort variable (refer to the Output Parameter sections) after any subprogram call. If Abort is set to one, the driver will inform the user that the specified model has been aborted, and the program will continue.

3. System Configuration :

Standard Memory Option

(Optional) Printer

(Optional) Graphics ROM and 1) CRT graphics hardware or
2) 9872A

(Optional) Internal Thermal Printer for CRT graphics dumps

```
2100 SUB Linear(X(*),Y(*),N,A,B,Regss,Resss,Totalss,Regms,Resms,F,Dfre
g,Dfres,Dftot,Abort)
2110 ! *** MODEL: Y=A+B*X ***
2120 ON ERROR GOTO Bomb
2130 Abort=0
2140 Y1=X1=Z=X2=Y2=0
2150 FOR I=1 TO N
2160    X1=X1+X(I)                  ! Sum of X's
2170    Y1=Y1+Y(I)                  ! Sum of Y's
2180    X2=X2+X(I)*X(I)             ! Sum of X squares
2190    Y2=Y2+Y(I)*Y(I)             ! Sum of Y squares
2200    Z=Z+X(I)*Y(I)               ! Sum of XY's
2210 NEXT I
2220 X1=X1/N                        ! Mean X
2230 Y1=Y1/N                        ! Mean Y
2240 B=(Z-N*X1*Y1)/(X2-N*X1*X1)
2250 A=Y1-B*X1
2260 Totalss=Y2-N*Y1*Y1             ! Total Sum of Squares
2270 Regss=(Z-N*X1*Y1)^2/(X2-N*X1*X1)! Regression Sum of Squares
2280 Resss=Totalss-Regss            ! Residual Sum of Squares
2290 Regms=Regss                    ! Regression Mean Squares
2300 Resms=Resss/(N-2)              ! Residual Mean Squares
2310 OFF ERROR
2320 DEFAULT ON
2330 F=Regms/Resms                  ! F Ratio
2340 DEFAULT OFF
2350 Dfreg=1                        ! Degrees of Freedom
2360 Dfres=N-2
2370 Dftot=Dfreg+Dfres
2380 SUBEXIT
2390 Bomb: Abort=1
2400 SUBEND
2410 SUB Logarithmic(X(*),Y(*),N,A,B,Regss,Resss,Totalss,Regms,Resms,F
,Dfreg,Dfres,Dftot,Abort)
2420 ! *** MODEL: Y=A+B*LN(X) ***
2430 ON ERROR GOTO Bomb
2440 Abort=0
2450 X1=Y1=Z=X2=Y2=0
2460 FOR I=1 TO N
2470    Logx=LOG(X(I))
2480    X1=X1+Logx                  ! Sum of X's
2490    Y1=Y1+Y(I)                  ! Sum of Y's
2500    X2=X2+Logx*Logx             ! Sum of X squares
2510    Y2=Y2+Y(I)*Y(I)             ! Sum of Y squares
2520    Z=Z+Logx*Y(I)               ! Sum of XY's
2530 NEXT I
2540 X1=X1/N                        ! Mean X
```

```
2550 Y1=Y1/N                          ! Mean Y
2560 B=(Z-N*X1*Y1)/(X2-N*X1*X1)
2570 A=Y1-B*X1
2580 Totalss=Y2-N*Y1*Y1               ! Total Sum of Squares
2590 Regss=(Z-N*X1*Y1)^2/(X2-N*X1*X1)! Regression Sum of Squares
2600 Resss=Totalss-Regss              ! Residual Sum of Squares
2610 Regms=Regss                      ! Regression Mean Squares
2620 Resms=Resss/(N-2)                ! Residual Mean Squares
2630 OFF ERROR
2640 DEFAULT ON
2650 F=Regms/Resms                    ! F Ratio
2660 DEFAULT OFF
2670 Dfreg=1                          ! Degrees of Freedom
2680 Dfres=N-2
2690 Dftot=Dfreg+Dfres
2700 SUBEXIT
2710 Bomb: Abort=1
2720 SUBEND
2730 SUB Exponential(X(*),Y(*),N,A,B,Regss,Resss,Totalss,Regms,Resms,F
,Dfreg,Dfres,Dftot,Abort)
2740 ! *** MODEL: Y=A*EXP(B*X) ***
2750 ON ERROR GOTO Bomb
2760 Abort=0
2770 X1=X2=Z=Y1=Y2=0
2780 FOR I=1 TO N
2790    Logy=LOG(Y(I))
2800    X1=X1+X(I)                    ! Sum of X's
2810    X2=X2+X(I)*X(I)               ! Sum of X squares
2820    Y1=Y1+Logy                    ! Sum of Y's
2830    Y2=Y2+Logy*Logy               ! Sum of Y squares
2840    Z=Z+X(I)*Logy                 ! Sum of XY's
2850 NEXT I
2860 X1=X1/N                          ! Mean X
2870 Y1=Y1/N                          ! Mean Y
2880 B=(Z-N*X1*Y1)/(X2-N*X1*X1)
2890 A=EXP(Y1-B*X1)
2900 Totalss=Y2-N*Y1*Y1               ! Total Sum of Squares
2910 Regss=(Z-N*X1*Y1)^2/(X2-N*X1*X1)! Regression Sum of Squares
2920 Resss=Totalss-Regss              ! Residual Sum of Squares
2930 Regms=Regss                      ! Regression Mean Squares
2940 Resms=Resss/(N-2)                ! Residual Mean Squares
2950 OFF ERROR
2960 DEFAULT ON
2970 F=Regms/Resms                    ! F Ratio
2980 DEFAULT OFF
2990 Dfreg=1                          ! Degrees of Freedom
3000 Dfres=N-2
```

```
3010 Dftot=Dfreg+Dfres
3020 SUBEXIT
3030 Bomb: Abort=1
3040 SUBEND
3050 SUB Polynomial(X(*),Y(*),N,Degree,Coeffs(*),Regss,Resss,Totalss,R
egms,Resms,F,Dfreg,Dfres,Dftot,Abort)
3060 ON ERROR GOTO Bomb
3070 Abort=0
3080 OPTION BASE 0
3090 DIM Matrix(Degree,Degree),Inv(Degree,Degree),B(Degree)
3100 REDIM Coeffs(Degree)
3110 IF Degree>N-2 THEN SUBEXIT  ! Check for higher degree than possib
le
3120 Dfreg=Degree
3130 Dfres=N-1-Degree
3140 Dftot=Dfreg+Dfres
3150 FOR K=0 TO Degree           !  Set up system of equations
3160    FOR J=K TO Degree
3170       Matrix(K,J)=0
3180       FOR I=1 TO N
3190          Matrix(K,J)=Matrix(K,J)+FNG(K)*FNG(J)
3200       NEXT I
3210    Matrix(J,K)=Matrix(K,J)
3220    NEXT J
3230    B(K)=0
3240    FOR I=1 TO N
3250       B(K)=B(K)+Y(I)*FNG(K)
3260    NEXT I
3270 NEXT K
3280 MAT Inv=INV(Matrix)         !  Solve the system of equations
3290 MAT Coeffs=Inv*B
3300 FOR I=1 TO N
3310  X1=X1+X(I)
3320  X2=X2+X(I)*X(I)
3330  Y1=Y1+Y(I)
3340  Y2=Y2+Y(I)*Y(I)
3350  Z=Z+X(I)*Y(I)
3360 NEXT I
3370 Y1=Y1/N
3380 X1=X1/N
3390 Totalss=Y2-N*Y1*Y1            ! Total Sum of Squares
3400 GOSUB Regss                   ! Regression Sum of Squares
3410 Resss=Totalss-Regss           ! Residual Sum of Squares
3420 Regms=Regss/Dfreg
3430 Resms=Resss/Dfres
3440 OFF ERROR
3450 DEFAULT ON
```

```
3460 F=Regms/Resms
3470 DEFAULT OFF
3480 SUBEXIT
3490 Regss: Regss=0
3500 FOR I=1 TO N
3510    J=0
3520    FOR L=0 TO Degree
3530       J=J+X(I)^L*Coeffs(L)
3540    NEXT L
3550    Regss=Regss+(J-Y1)^2
3560 NEXT I
3570 RETURN
3580 SUBEXIT
3590 DEF FNG(M)=X(I)^M
3600 Bomb: Abort=1
3610 SUBEND
```

Driver Utilization

File Name: REGD

This driver allows the user to enter data from the keyboard (or use canned data from the Utility Library for demo purposes) and then select any or all of the available types of curves to fit to the data. The driver prints out the coefficients of the computed curves, it prints an analysis of variance table (AOV table), and it allows the user the option of plotting the data points and the curves. The plotting routine is loaded only after the user indicates that he wants his results plotted. The plotting routine is not necessary to run the program.

User Instructions:

1. Insert the Utility Routines cartridge 2 into the primary transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "REGD:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Printer select code?" appears in the display area:
   a. Enter: The select code of the printer (Note: The select code of the CRT is 16 and is the default value. The select code of the internal printer is Ø, if your machine has an internal printer.)
   b. Press: CONT
5. When "Do you want your results plotted (Y/N)?" appears in the display area:
   a. If you want your results plotted:
      1) Type: Y
      2) Press: CONT
      3) Go to step 6
      or

a. If you do not want your results plotted:
   1) Type: N
   2) Press: CONT
   3) Go to step 7.

6. When "Please enter the number of the device you wish to use" appears in the display area:
   a. Enter: The number of the device you wish to use (Note: The available plotting devices are listed on the screen).
   b. Press: CONT
   c. If you entered 1 in step 6a, go to step 7. Otherwise, go to step 6d.
   d. When "Please enter the select code of the graphics device" appears in the display area:
      1) Enter: The select code of the specified plotting device.
      2) Press: CONT
   e. When "Please enter the HPIB address of the graphics device" appears in the display area:
      1) Enter: The HPIB address of the specified plotting device (If the device is not HPIB, enter -1).
      2) Press: CONT

7. When "Which mode do you prefer to use, manual or canned (M/C)?" appears in the display area:
   a. If you wish to enter the data manually:
      1) Type: M
      2) Press: CONT
      3) Go to step 8.

      or

   a. If you wish to use canned data:
      1) Type: C
      2) Press: CONT
   b. When "Which data file do you wish to use (1, 2, or 3)?" appears in the display area:

1) Enter: The number of the data file you wish to use.
2) Press: CONT
3) Go to step 12.

8. When "Please enter the number of points you have" appears in the display area of the CRT:
   a. Enter: The number of points you have.
   b. Press: CONT
9. When "Please enter the coordinates for point #i (X,Y, CONT)" appears in the display area:
   a. Enter: The X and Y coordinates of the ith point, separated by a comma.
   b. Press: CONT
   c. Repeat step 9 as often as necessary.
10. When "Enter the subscript of the point you wish to change" appears in the display area:
   a. If you wish to make any changes:
      1) Enter: The subscript of the point you wish to change.
      2) Press: CONT
      3) Go to step 11.

      or

   a. If you do not wish to make any changes:
      1) Press: CONT
      2) Go to step 12.
11. When "Please enter the coordinates for point # i (X,Y, CONT)?" appears in the display area:
   a. Enter: The X and Y coordinates of the ith point, separated by a comma.
   b. Press: CONT
   c. Go to step 10.
12. When "Please select the model you wish to use" appears in the display area of the CRT:
   a. If you want the program to fit a straight line to the data points (equation: Y=A+B*X):
      1) Enter: 1
      2) Press: CONT
      3) Repeat step 12.

b. If you want the program to fit a logarithmic curve to the data points (equation: Y=A+B*LOG(X)))
   1) Enter: 2
   2) Press: CONT
   3) Repeat step 12.

c. If you want the program to fit an exponential curve to the data points (equation: Y=A*EXP(B*X))
   1) Enter: 3
   2) Press: CONT
   3) Repeat step 12.

d. If you want the program to fit a polynomial curve to the data points (equation: Y=A(N)*X^N+A(N-1)*X^(N-1)+...+A(1)* X+A(0))
   1) Enter: 4
   2) Press: CONT
   3) When "Please enter the degree of the polynomial you want to fit to the points" appears in the display area:
      a) Enter: The degree of the polynomial you wish to use.
      b) Press: CONT
      c) Repeat step 12.

e. If you want the program to use all of the available models:
   1) Enter: 5
   2) Press: CONT
   3) Wait for the linear, log, and exponential curves to be computed then go to step 12d 3).

f. If you want to end the program:
   1) Enter: 6
   2) Press: CONT
   3) The program stops.

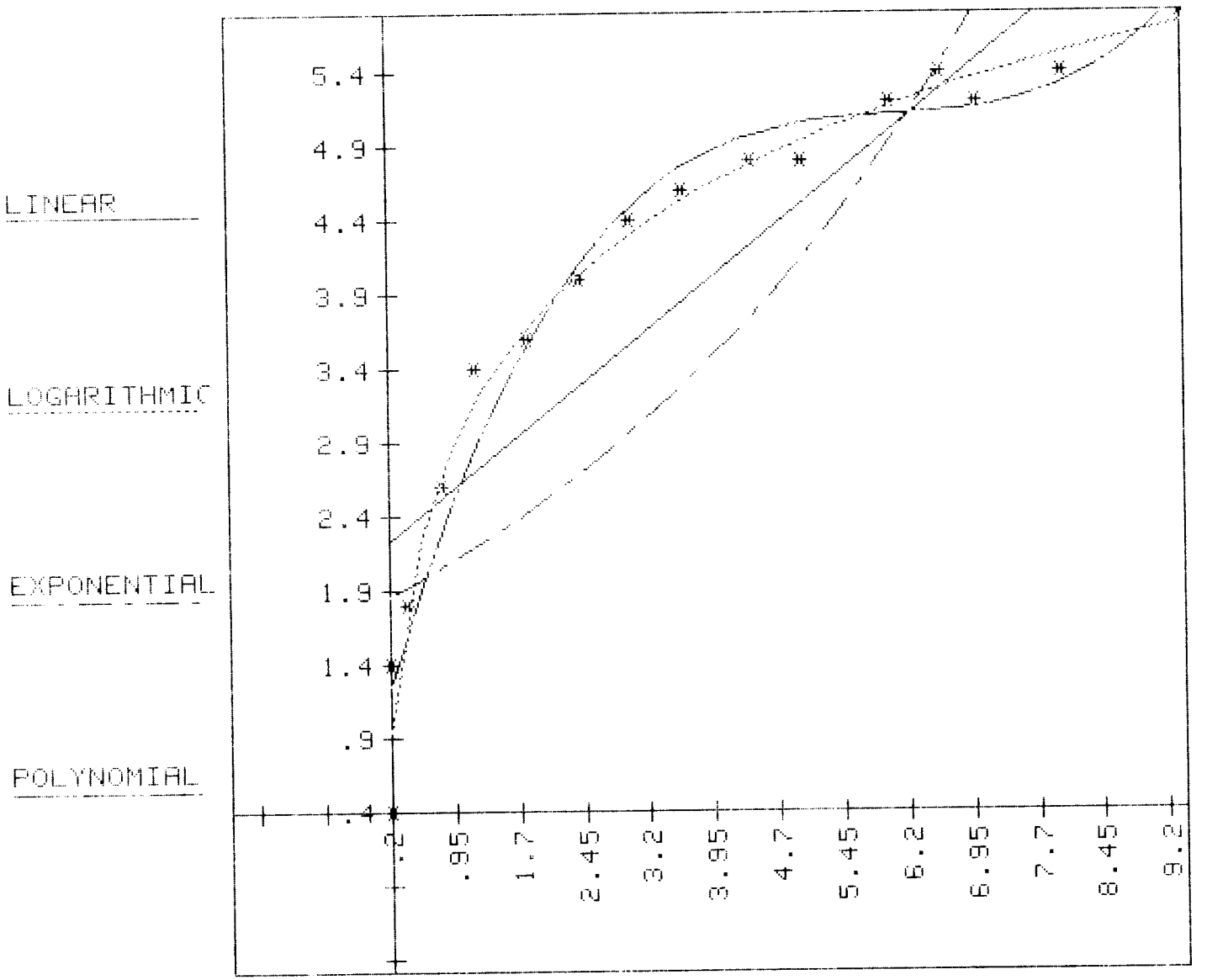
```
            DATA
   Point #1:      X=.2      Y=.4
   Point #2:      X=.2      Y=1.4
   Point #3:      X=.4      Y=1.8
   Point #4:      X=.8      Y=2.6
   Point #5:      X=1.2      Y=3.4
   Point #6:      X=1.8      Y=3.6
   Point #7:      X=2.4      Y=4
   Point #8:      X=3      Y=4.4
   Point #9:      X=3.6      Y=4.6
  Point #10:      X=4.4      Y=4.8
  Point #11:      X=5      Y=4.8
  Point #12:      X=6      Y=5.2
  Point #13:      X=6.6      Y=5.4
  Point #14:      X=7      Y=5.2
  Point #15:      X=8      Y=5.4
  Point #16:      X=9.4      Y=5.8


Linear model: Y=A+B*X
A= 2.13760107817
B= .476639712489

Source          Df            SS            MS            F

Regression       1          30.343        30.343        50.289
Residual        14           8.447          .603
Total           15          38.790


Logarithmic Model: Y=A+B*LN(X)
A= 2.93792027118
B= 1.24329619278

Source          Df            SS            MS            F

Regression       1          38.129        38.129       807.137
Residual        14            .661          .047
Total           15          38.790
```

```
Exponential Model: Y=A*EXP(B*X)
A= 1.79580472315
B= .167668287426
```

| Source | Df | SS | MS | F |
|---|---|---|---|---|
| Regression | 1 | 3.755 | 3.755 | 14.654 |
| Residual | 14 | 3.587 | .256 | |
| Total | 15 | 7.342 | | |

```
POLYNOMIAL MODEL: Y=A(M)*X^M+A(M-1)*X^(M-1)+...+A(1)*X+A(0)
Coefficients:
   A(0)=.8487020449
   A(1)=2.045048134
   A(2)=-.3324347981
   A(3)=.01832368985
```

| Source | Df | SS | MS | F |
|---|---|---|---|---|
| Regression | 3 | 37.382 | 12.461 | 106.160 |
| Residual | 12 | 1.408 | .117 | |
| Total | 15 | 38.790 | | |


LINEAR
LOGARITHMIC
EXPONENTIAL
POLYNOMIAL
5.4
4.9
4.4
3.9
3.4
2.9
2.4
1.9
1.4
.9
.4
.2
.95
1.7
2.45
3.2
3.95
4.7
5.45
6.2
6.95
7.7
8.45
9.2

```
10   PRINTER IS 16
20   LINK "REGS",2150
30   PRINT PAGE,"Any instructions or prompts that occur while this pro
gram is "
40   PRINT "running will appear on the CRT only.  The output for the p
rogram"
50   PRINT "will be printed on the device having the select code you a
re"
60   PRINT "asked to enter below.  If you press CONTINUE without enter
ing"
70   PRINT "a number, the output will appear on the default printer, t
he CRT"
80   PRINT "(select code 16).",LIN(4)
90   Select=16
100  INPUT "Printer select code?",Select
110  Select=INT(Select)
120  IF Select<0 THEN 90
130  PRINT PAGE;"If you have graphics available on this machine, you c
an have"
140  PRINT "your data plotted, as well as the curves you fit to it."
150  INPUT "Do you want your results plotted (Y/N)?",A$
160  IF UPC$(A$)="Y" THEN 220
170  IF UPC$(A$)="N" THEN 200
180  BEEP
190  GOTO 150
200  Graphics=0
210  GOTO 380
220  Graphics=1
230  LINK "REGPLT",5000
240  PRINT PAGE;"The available devices are listed below.  Please selec
t the number"
250  PRINT "of the device on which you wish to plot your data.",LIN(1)
260  PRINT SPA(5);"1.  CRT"
270  PRINT SPA(5);"2.  9872A"
280  INPUT "Please enter the number of the device you wish to use",Whi
chdevice
290  Whichdevice=INT(Whichdevice)
300  IF (Whichdevice>=1) AND (Whichdevice<=2) THEN 350
310  BEEP
320  DISP "Device not available"
330  WAIT 700
340  GOTO 280
350  IF Whichdevice=1 THEN 380
360  INPUT "Please enter the select code of the graphics device",Pselе
ct
370  INPUT "Please enter the HPIB address of the graphics device",Hpib
380  PRINT PAGE;"There are two modes of input for this program.  You m
```

```
ay enter"
390  PRINT "your own data, or you may use canned data (to familiarize
yourself"
400  PRINT "with the program).  Below you are asked to select which mo
de of input"
410  PRINT "you prefer to use.  Enter M for manual or C for canned.  I
f you press"
420  PRINT "CONT without entering anything, M will be the assumed answ
er."
430  A$="M"
440  INPUT "Which mode do you prefer to use, manual or canned (M/C)?",
A$
450  IF (A$="m") OR (A$="M") THEN Manual
460  IF (A$="c") OR (A$="C") THEN Auto
470  GOTO 430
480 Manual:   PRINT PAGE
490           X=1
500           INPUT "Please enter the number of points you have",N
510           N=INT(N)
520           IF N>2 THEN Okay
530           PRINT PAGE,"You must enter at least three points.",LIN(2
)
540           GOTO 500
550 Auto:  PRINT PAGE
560        N=16
570        X=0
580 Okay:  CALL Driver(N,Select,X,Graphics,Whichdevice,Pselect,Hpib)
590  END
600  SUB Driver(N,Select,Input,Graphics,Whichdevice,Pselect,Hpib)
610  OPTION BASE 1
620  DIM X(N),Y(N),A$[2],B$[2],Coeffs(0:10)
630  IF Input THEN Manual
640 Canned:  File=PI
650          INPUT "Which data file do you wish to use (1,2, or 3)?",F
ile
660          IF File=PI THEN Canned
670          File=INT(File)
680          IF (File<1) OR (File>3) THEN Canned
690          A$="X"&VAL$(File)
700          B$="Y"&VAL$(File)
710          ASSIGN #1 TO A$
720          ASSIGN #2 TO B$
730          READ #1;X(*)
740          READ #2;Y(*)
750          GOSUB Print
760          GOTO Run
770 Manual: PRINT PAGE;"Please enter both the X an Y coordinates for a
```

```
 point,"
780          PRINT "seperated by a comma, as the proper prompt appears
in the display"
790          PRINT "area.  If you make any mistakes while keying in the
 data, you"
800          PRINT "will be given the chance to make corrections later.
"
810          FOR I=1 TO N
820             DISP "Please enter the coordinates for point #";I;"(X,Y
,CONT)";
830             INPUT "",X(I),Y(I)
840             PRINT USING Image;I,X(I),Y(I)
850          NEXT I
860          GOSUB Print
870          IF Select<>16 THEN PRINT PAGE;
880          PRINT "If you want to make any changes, enter the subcript
 of the "
890          PRINT "point you wish to change.  Otherwise, press CONT wi
thout"
900          PRINT "entering anything.",LIN(2)
910          PRINTER IS Select
920 Editions: I=PI
930          INPUT "Enter the subscript of the point you wish to change
",I
940          IF I=PI THEN Run
950          I=INT(I)
960          IF (I<1) OR (I>N) THEN Editions
970          DISP "Please enter the coordinates for point #";I;" (X,Y,C
ONT)";
980          INPUT "",X(I),Y(I)
990          PRINT USING Image;I,X(I),Y(I)
1000         GOTO Editions
1010 Run:PRINTER IS 16
1020      IF Graphics THEN CALL Plot(0,A,B,Coeffs(*),X(*),Y(*),N,Degree
,Whichdevice,Pselect,Hpib,Select)
1030      IF Select<>16 THEN PRINT PAGE;
1040      PRINT "Please select the model of curve fit you wish to use.
 The"
1050      PRINT "following models are available:"
1060      PRINT SPA(5);"1=Linear Model (Y=A+B*X)"
1070      PRINT SPA(5);"2=Logarithmic Model (Y=A+B*LN(X))"
1080      PRINT SPA(5);"3=Exponential Model (Y=A*EXP(B*X))"
1090      PRINT SPA(5);"4=Polynomial Model (Y=A(N)*X^N+A(N-1)*X^(N-1)+.
..+A(0))"
1100      PRINT SPA(5);"5=All of the above"
1110      PRINT SPA(5);"6=End",LIN(1)
1120      Flag=0
```

```
1130      PRINTER IS Select
1140      INPUT "Please select the model you wish to use",Q
1150      Q=INT(Q)
1160      IF (Q>=1) AND (Q<=6) THEN 1210
1170      BEEP
1180      DISP "INVALID REQUEST"
1190      WAIT 700
1200      GOTO 1140
1210      IF Q<6 THEN Around
1220      BEEP
1230      PRINTER IS 16
1240      IF Select<>16 THEN PRINT PAGE
1250      DISP "PROGRAM DONE"
1260      SUBEXIT
1270 Around:IF Q<5 THEN Single
1280      Flag=1                          ! Flag=1 -> do all models
1290      Q=1
1300 Single:ON Q GOTO Li,Lo,E,P
1310 Li: CALL Linear(X(*),Y(*),N,A,B,Regss,Resss,Totalss,Regms,Resms,F
,Dfreg,Dfres,Dftot,Abort)
1320      IF NOT Abort THEN 1380
1330      BEEP
1340      DISP "Linear Model Aborted"
1350      PRINT "Linear Model Aborted",LIN(1)
1360      WAIT 1000
1370      GOTO 1400
1380      PRINT "Linear model: Y=A+B*X"
1390      GOSUB Printout
1400      IF NOT Flag THEN 1140
1410      Q=Q+1
1420 Lo: CALL Logarithmic(X(*),Y(*),N,A,B,Regss,Resss,Totalss,Regms,Re
sms,F,Dfreg,Dfres,Dftot,Abort)
1430      IF NOT Abort THEN 1490
1440      BEEP
1450      DISP "Logarithmic Model Aborted"
1460      PRINT "Logarithmic Model Aborted",LIN(1)
1470      WAIT 1000
1480      GOTO 1510
1490      PRINT "Logarithmic Model: Y=A+B*LN(X)"
1500      GOSUB Printout
1510      IF NOT Flag THEN 1140
1520      Q=Q+1
1530 E:   CALL Exponential(X(*),Y(*),N,A,B,Regss,Resss,Totalss,Regms,Re
sms,F,Dfreg,Dfres,Dftot,Abort)
1540      IF NOT Abort THEN 1600
1550      BEEP
1560      DISP "Exponential Model Aborted"
```

```
1570      PRINT "Exponential Model Aborted",LIN(1)
1580      WAIT 1000
1590      GOTO 1620
1600      PRINT "Exponential Model: Y=A*EXP(B*X)"
1610      GOSUB Printout
1620      IF NOT Flag THEN 1140
1630      Q=Q+1
1640      GOTO P
1650 Printout: PRINT "A=";A
1660      PRINT "B=";B,LIN(2)
1670      GOSUB Aovtable
1680      IF Graphics AND (Select=16) THEN WAIT 2000
1690      IF Graphics THEN CALL Plot(Q,A,B,Coeffs(*),X(*),Y(*),N,Degree
,Whichdevice,Pselect,Hpib,Select)
1700      RETURN
1710 P:   INPUT "Please enter the degree of the polynomial you wish to
fit to the points",Degree
1720      Degree=INT(Degree)
1730      IF Degree<=N-2 THEN Call
1740      BEEP
1750      DISP "The degree of the polynomial may be no greater than "&V
AL$(N-2)&".  Try again."
1760      WAIT 1500
1770      GOTO P
1780 Call:  CALL Polynomial(X(*),Y(*),N,Degree,Coeffs(*),Regss,Resss,T
otalss,Regms,Resms,F,Dfreg,Dfres,Dftot,Abort)
1790      IF NOT Abort THEN 1850
1800      BEEP
1810      DISP "Polynomial Model Aborted"
1820      PRINT "Polynomial Model Aborted",LIN(1)
1830      WAIT 1000
1840      GOTO 1950
1850      PRINT "POLYNOMIAL MODEL: Y=A(M)*X^M+A(M-1)*X^(M-1)+...+A(1)*X
+A(0)"
1860      PRINT "Coefficients:"
1870      FOR I=0 TO Degree
1880         PRINT USING Set;I,Coeffs(I)
1890 Set:    IMAGE "A("DDDD")=",K
1900      NEXT I
1910      PRINT LIN(1)
1920      GOSUB Aovtable
1930      IF Graphics AND (Select=16) THEN WAIT 2000
1940      IF Graphics THEN CALL Plot(Q,A,B,Coeffs(*),X(*),Y(*),N,Degree
,Whichdevice,Pselect,Hpib,Select)
1950      Flag=0
1960      GOTO 1140
1970 Print: PRINTER IS Select
```

```
1980          PRINT LIN(2),SPA(12);"DATA"
1990          FOR I=1 TO N
2000             PRINT USING Image;I,X(I),Y(I)
2010 Image:      IMAGE "Point #"DDDD":",5X,"X="K,5X,"Y="K
2020          NEXT I
2030          PRINT LIN(2)
2040          PRINTER IS 16
2050          RETURN
2060 Aovtable: PRINT USING 2070
2070          IMAGE "Source     "5X" Df"11X" SS"13X" MS"12X"F",/
2080          PRINT USING 2090;Dfreg,Regss,Regms,F
2090          IMAGE "Regression"5X,MDD,4X,M7D.3D,4X,M7D.3D,5X,M4D.3D
2100          PRINT USING 2110;Dfres,Resss,Resms
2110          IMAGE "Residual   "5X,MDD,4X,M7D.3D,4X,M7D.3D
2120          PRINT USING 2130;Dftot,Totalss
2130          IMAGE "Total      "5X,MDD,4X,M7D.3D,/,/,/,/,/
2140          RETURN
2150          END
```

# Listing of Plotting Routine for Family Regression

```
5000 SUB Plot(Q,A,B,Coeffs(*),X(*),Y(*),N,Degree,Whichdevice,Pselect,H
pib,Select)
5010          IF Whichdevice=1 THEN GRAPHICS
5020          DATA -2,-1,1,2
5030          READ Um,Dm,Md,Mu
5040          DATA .39794,.69897,.87506
5050          READ Log2,Log5,Log7
5060          Xmin=FNMin(X(*),N)
5070          Xmax=FNMax(X(*),N)
5080          Lx=LGT(Xmax-Xmin)
5090          Ymin=FNMin(Y(*),N)
5100          Ymax=FNMax(Y(*),N)
5110          Ly=LGT(Ymax-Ymin)
5120          Xfudge=.20*(Xmax-Xmin)
5130          Yfudge=.20*(Ymax-Ymin)
5140 ON Q+1 GOTO Setup,Linear,Logarithmic,Exponential,Polynomial
5150 Setup:   ON Whichdevice GOTO Crt,P9872
5160 Crt:     PLOTTER IS "GRAPHICS"
5170          GOTO 5190
5180 P9872:   PLOTTER IS Pselect,Hpib,"9872A"
5190          GCLEAR
5200          GRAPHICS
5210          LOCATE 23,123,0,100
5220          FRAME
5230          LINE TYPE 1
5240          SCALE Xmin-Xfudge,Xmax,Ymin-Yfudge,Ymax
5250          Testxtic=FRACT(Lx)+(Lx<0)
5260          Testytic=FRACT(Ly)+(Ly<0)
5270          Xtic=10^(INT(Lx)-1)*(1+1.5*((Testxtic>Log2) AND (Testxtic
<Log5))+4*((Testxtic>=Log5) AND (Testxtic<=Log7))+6.5*(Testxtic>Log7))
5280          Ytic=10^(INT(Ly)-1)*(1+1.5*((Testytic>Log2) AND (Testytic
<Log5))+4*((Testytic>=Log5) AND (Testytic<=Log7))+6.5*(Testytic>Log7))
5290          CALL Laxes(Xtic,Ytic,Xmin,Ymin,1,1,2,Xmin-Xfudge,Xmax,Ymi
n-Yfudge,Ymax)
5300          LORG 5
5310          FOR I=1 TO N
5320             MOVE X(I),Y(I)
5330             LABEL USING 5340;"*"
5340             IMAGE A
5350          NEXT I
5360          LORG 1
5370          PENUP
5371          IF Select<>16 THEN 5400
5380          WAIT 1500
5390          EXIT GRAPHICS
5400          SUBEXIT
5410 Linear:  DEF FNLinear(X)=A+B*X
```

```
5420          SETGU
5430          CLIP 0,123,0,100
5440          LINE TYPE 1
5450          MOVE 0,80
5460          LABEL USING 5470;"LINEAR"
5470          IMAGE K
5480          MOVE 0,79
5490          LINE TYPE 1
5500          IPLOT 20,0,Dm
5510          CLIP 23,123,0,100
5520          SETUU
5530          PENUP
5540          FOR I=Xmin TO Xmax+Xfudge STEP (Xmax+Xfudge-Xmin)/2
5550             PLOT I,FNLinear(I),Md
5560          NEXT I
5570          PENUP
5571          IF Select<>16 THEN 5600
5580          WAIT 1500
5590          EXIT GRAPHICS
5600          SUBEXIT
5610 Logarithmic:  DEF FNLog(X)=A+B*LOG(X)
5620          SETGU
5630          CLIP 0,123,0,100
5640          LINE TYPE 1
5650          MOVE 0,60
5660          LABEL USING 5470;"LOGARITHMIC"
5670          LINE TYPE 3
5680          MOVE 0,59
5690          IPLOT 20,0,Dm
5700          PENUP
5710          CLIP 23,123,0,100
5720          SETUU
5730          FOR I=Xmin TO Xmax+Xfudge STEP (Xmax+Xfudge-Xmin)/30
5740            PLOT I,FNLog(I),Md
5750          NEXT I
5760          PENUP
5761          IF Select<>16 THEN 5790
5770          WAIT 1500
5780          EXIT GRAPHICS
5790          SUBEXIT
5800 Exponential:  DEF FNExp(X)=A*EXP(B*X)
5810          SETGU
5820          CLIP 0,123,0,100
5830          LINE TYPE 1
5840          MOVE 0,40
5850          LABEL USING 5470;"EXPONENTIAL"
5860          LINE TYPE 4
```

```
5870          MOVE 0,39
5880          IPLOT 20,0,Dm
5890          CLIP 23,123,0,100
5900          SETUU
5910          PENUP
5920          FOR I=Xmin TO Xmax+Xfudge STEP (Xmax+Xfudge-Xmin)/30
5930            PLOT I,FNExp(I),Md
5940          NEXT I
5950          PENUP
5951          IF Select<>16 THEN 5980
5960          WAIT 1500
5970          EXIT GRAPHICS
5980          SUBEXIT
5990 Polynomial:  !
6000          SETGU
6010          CLIP 0,123,0,100
6020          LINE TYPE 1
6030          MOVE 0,20
6040          LABEL USING 5470;"POLYNOMIAL"
6050          LINE TYPE 6
6060          MOVE 0,19
6070          IPLOT 20,0,Dm
6080          PENUP
6090          CLIP 23,123,0,100
6100          SETUU
6110          FOR I=Xmin TO Xmax+Xfudge STEP (Xmax+Xfudge-Xmin)/30
6120             J=0
6130             FOR L=0 TO Degree
6140                J=J+I^L*Coeffs(L)
6150             NEXT L
6160             PLOT I,J,Md
6170          NEXT I
6180          PENUP
6181          IF Select<>16 THEN 6210
6190          WAIT 1500
6200          EXIT GRAPHICS
6210          SUBEXIT
6220 DEF FNMax(X(*),N)
6230 X=X(1)
6240 FOR I=2 TO N
6250 X=MAX(X,X(I))
6260 NEXT I
6270 RETURN X
6280 DEF FNMin(X(*),N)
6290 X=X(1)
6300 FOR I=2 TO N
6310 X=MIN(X,X(I))
```

```
6320 NEXT I
6330 RETURN X
6340 SUB Laxes(Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize,Xmin,Xmax,Ymin
,Ymax)
6350 IF (Xmin)=Xmax) OR (Ymin)=Ymax) THEN SUBEXIT
6360 Xfudge=.02*(Xmax-Xmin)
6370 Yfudge=.02*(Ymax-Ymin)
6380 AXES Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize
6390 DEG
6391 IF NOT Xtic THEN Labely
6400 Labelx:IF SGN(Xtic)=-1 THEN Parx
6410 LDIR 90
6420 LORG 8
6430 GOTO 6460
6440 Parx: LDIR 0
6450 LORG 6
6460 FOR I=Xorg TO Xmax STEP ABS(Xtic)
6470 MOVE I,Yorg-Yfudge
6480 LABEL USING 6490;I
6490 IMAGE #,K
6500 NEXT I
6550 Labely: IF NOT Ytic THEN SUBEXIT
6555 IF SGN(Ytic)=-1 THEN Pary
6560 LDIR 0
6570 LORG 8
6580 GOTO 6610
6590 Pary: LDIR -90
6600 LORG 6
6610 FOR I=Yorg TO Ymax STEP ABS(Ytic)
6620 MOVE Xorg-Xfudge,I
6630 LABEL USING 6490;I
6640 NEXT I
6690 SUBEXIT
```

Subprogram Name: Pie

This subprogram plots pie charts on the CRT. The chart includes raw numbers,percentages, titles, and labels.

Subprogram Utilization:

File Name: Pie
Calling Syntax: CALL Pie (A(*), L$(*), N, Units$, Title$)

Input Parameters:

A(*) - This one-dimensional full-precision numeric array contains the values of each data cell

L$(*) - This one-dimensional string array contains the titles of each data cell

N - This full-precision variable, constant, or expression tells how many data cells there are

Units$ - This string holds the units that the numbers are in (used for titling).

Title$ - This string is a title for the chart

Local Variables:

A - Temporary variable for drawing the pies

A(*) - Holds the values of each data cell

Angle - Temporary variable for slicing the pies

B - Temporary variable for drawing the pies

Cos - The cosine of 8°(used for drawing the pies)

I - Loop counter

Inc - Temporary variable for slicing the pies

L$(*) - Holds the titles of each data cell

| | | |
|---|---|---|
| Letter$ | - | Indexing string |
| Linc | - | Distance (in GDU's) between each data cell's title |
| N | - | The number of data cells |
| S | - | The sum of the data cells |
| Sin | - | The sine of 8° (used for drawing the pies) |
| T | - | Loop counter |
| Title$ | - | The title of the plot |
| Units$ | - | The units of the data cells |
| X | - | Temporary variable for drawing the pies<br>Also used in placing labels |
| Y | - | Temporary variable for drawing the pies<br>Also used in placing labels |

Special Considerations and Programming Hints:

1. If the labels of the data cells run off the right edge of the screen, make them shorter.
2. If the charts are hard to read due to too many significant digits, or too many trailing zeroes, better results may be obtained by specifying the order of magnitude in the Units$ string (see the section under Input Parameters) and then entering 2 or 3 digit numbers for the data cells.
3. The provided driver (file "BARPIE") will drive either the bar graphs subprogram, or the pie charts subprogram, or both.
4. System Configuration:

   Standard Memory Option
   (Optional) Internal Thermal Printer
   CRT graphics hardware
   Graphics ROM

## Annotated Listing of Subprogram Pie

```
6000 SUB Pie(A(*),L$(*),N,Units$,Title$)
6010 DIM Letter$[26]
6020 Letter$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
6030 PLOTTER IS "GRAPHICS"
6040 LOCATE 0,50,0,50
6050 LORG 5
6060 MOVE 25,25
6070 LABEL USING 6080;Title$
6080 IMAGE #,K
6090 LOCATE 0,50,50,100
6100 SCALE -1.5,1.5,-1.5,1.5
6110 GRAPHICS
6120 PEN 1
6130 S=0
6140 FOR I=1 TO N       ! Sum the data together
6150 S=S+A(I)
6160 NEXT I
6170 DEG
6180 T=1
6190 Cos=COS(8)
6200 Sin=-SIN(8)
6210 X=1
6220 Y=0
6230 MOVE X,Y
6240 FOR I=8 TO 360 STEP 8    ! Draw the pie
6250 A=X*Cos+Y*Sin
6260 B=-X*Sin+Y*Cos
6270 DRAW A,B
6280 X=A
6290 Y=B
6300 NEXT I
6310 DRAW 0,0
6320 Angle=0
6330 LORG 5
6340 FOR I=1 TO N             ! Slice the pie and label it
6350 Inc=A(I)/S*360
6360 Angle=Angle+Inc
6370 DRAW COS(Angle),SIN(Angle)
6380 MOVE COS(Angle-.5*Inc)*.75,SIN(Angle-.5*Inc)*.75
6390 LABEL USING 6400;Letter$[I,I]
6400 IMAGE #,A,"."
6410 MOVE 0,0
6420 NEXT I
6430 ON T GOTO 6440,6480
6440 LOCATE 50,100,0,50
6450 SCALE -1.5,1.5,-1.5,1.5
6460 T=2
```

```
6470 GOTO 6210
6480 LOCATE 50,100,50,100
6490 CLIP 0,123,0,100
6500 SCALE 50,100,50,100
6510 MOVE 75,47
6520 LORG 5
6530 LABEL USING 6670;"PERCENTAGES"
6540 MOVE 25,53
6550 LABEL USING 6670;Units$
6560 Linc=50/(N+1)
6570 FOR I=1 TO N
6580 LORG 1
6590 MOVE 60,100-I*Linc
6600 LABEL USING 6610;Letter$[I,I],L$(I)
6610 IMAGE #,A".  ",K
6620 LORG 1
6630 POINTER 25,75
6640 DIGITIZE X,Y
6650 MOVE X,Y
6660 LABEL USING 6670;VAL$(A(I))
6670 IMAGE #,K
6680 POINTER 75,25
6690 DIGITIZE X,Y
6700 MOVE X,Y
6710 FIXED 1
6720 LABEL USING 6670;VAL$(A(I)/S*100)&"%"
6730 IMAGE #,K
6740 STANDARD
6750 NEXT I
6760 SUBEXIT
```

Driver Utilization:

File Name: BARPIE

This driver allows the user to enter the titles and values of as many data cells as he wants. He may edit the values and titles at any time. Then this data is displayed graphically in the form of either a bar graph or a pie chart, or both. In addition, the graphs may be dumped to the internal thermal printer for hardcopy if the user desires. (The dump is optional - it is not necessary to have the internal printer to run this program.)

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the program:
   a. Type: LOAD "BARPIE:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Which of the above choices do you want (1, 2, or 3)?" appears in the display area of the CRT:
   a. Enter: 1, 2, or 3, depending upon whether you want to use the Bar Graph, the Pie Chart, or both, respectively.
   b. Press: CONT
5. When "Do you want hardcopy (Y/N)?" appears in the display area:
   a. If you want to use the CRT/internal thermal printer dump feature:
      1) Type: Y
      2) Press: CONT

      or
   a. If you do not want to use the CRT/internal thermal printer dump feature:
      1) Type: N
      2) Press: CONT

6. When "How many categories are there?" appears in the display area of the CRT:
   a. Enter: The number of categories, or data cells, that you want to use.
   b. Press: CONT
7. Repeat this step as often as necessary.
   a. When "Please enter the title of category # i" appears in the display area:
      1) Type: The title of the ith category.
      2) Press: CONT
   b. When "Please enter the value for [*title*]" appears in the display area:
      1) Enter: The numeric value to be associated with the indicated category.
      2) Press: CONT
   c. Repeat step 7 as often as necessary.
8. When "Would you like to make any changes in the above data (Y/N)?" appears in the display area of the CRT:
   a. If you want to make any changes:
      1) Type: Y
      2) Press: CONT
      3) Go to step 9.

      or

   a. If you do not want to make any changes:
      1) Type: N
      2) Press: CONT
      3) Go to step 12.
9. When "Please enter the subscript of the item you wish to change" appears in the display:
   a. If you want to make a change:
      1) Enter: The subscript of the item you wish to change.
      2) Press: CONT
      3) Go to step 10.

      or

a. If you do not wish to make any changes:
    1) Press: CONT
    2) Go to step 12.

10. When "Make any changes in the title and press CONT" appears in the display area:
    a. Make any desired changes in the title.
    b. Press: CONT

11. When "Enter a new value for [*title*] or press CONT to keep the old one" appears in the display area:
    a. If you wish to change the old value of the indicated category:
        1) Enter: The new value
        2) Press: CONT
        3) Go to step 9.

        or

    a. If you do not wish to change the old value of the indicated category:
        1) Press: CONT
        2) Go to step 9.

12. When "What units are you using (dollars, hours, people, etc.)?" appears in the display area:
    a. Type: The units you are using
    b. Press: CONT

13. When "Please enter the title of the plot" appears in the display area:
    a. Type: The title of the plot
    b. Press: CONT

14. If you asked for a bar graph in step 4, the bar graph will be drawn, and a beep will sound. The graph will remain on the screen for as long as you want.
    a. To proceed with the program, press CONT.

15. If you asked for a pie chart in step 4, the pie chart will be drawn, and a pair of crosshairs will appear in the center of the pie in the upper left corner.
    a. Read the bottom line in the upper right hand quadrant of the CRT.

b. Using the DISPLAY keys on the 9845A keyboard, position the crosshairs in the upper left pie where you would like the value of the category described to be labelled.

c. Press: CONT

d. Now the crosshairs will appear in the center of the lower right pie. Using the DISPLAY keys, position the crosshairs in the lower right pie to where you would like the percentage value of the category described in part 15a to be labelled.

e. Press: CONT

f. Repeat steps 15a-15e as often as necessary.

g. A beep will sound upon completion of the pie chart. When you are ready for the program to proceed, press CONT.

16. When "Do you want to run again (Y/N)?" appears in the display area of the CRT:

a. If you want to run the program again:

1) Type: Y

2) Press: CONT

3) Go to step 17.

or

a. If you do not want to run the program again:

1) Type: N

2) Press: CONT

3) The program ends

17. When "Do you want to modify the old data, or use new data (O/N)?" appears in the display area:

a. If you want to use the old data:

1) Type: O

2) Press: CONT

3) Go to step 8.

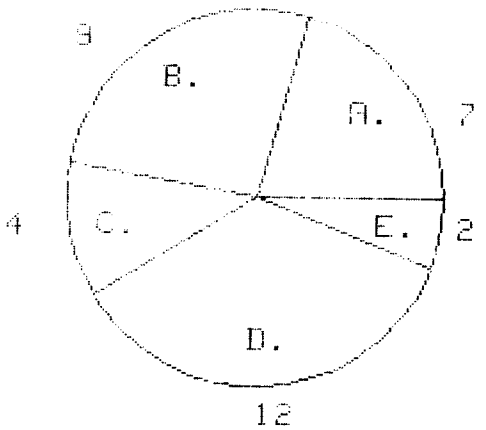
or

a. If you want to use new data:

1) Type: N

2) Press: CONT

3) Go to step 4.




Billions of Dollars

A. Marketing

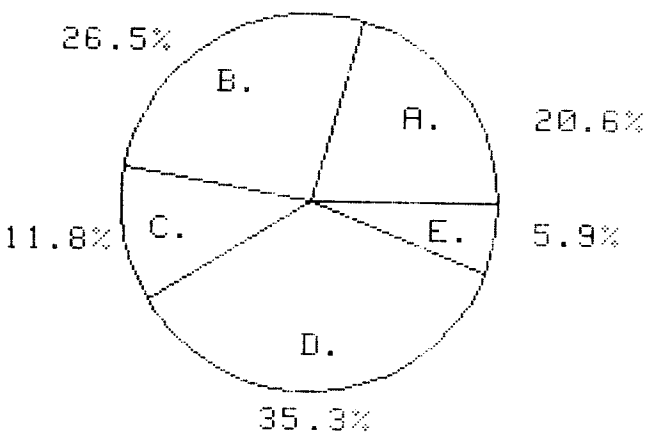
B. Research

C. Taxes

D. Manufacturing

E. Administration

PERCENTAGES




Corporate Expenses

```
10   PRINTER IS 16
20   PRINT PAGE;"   This program allows the user to enter some data an
d have a bar chart,"
30   PRINT "a pie chart, or both, computed and drawn.  Below you are a
sked to indicate"
40   PRINT "which choice you prefer.",LIN(1)
50   PRINT SPA(10);"1 -- Bar Chart"
60   PRINT SPA(10);"2 -- Pie Chart"
70   PRINT SPA(10);"3 -- Both of the above"
80   PRINT LIN(1);"(After each chart is drawn, a beep will sound.  The
 chart will"
90   PRINT "remain on the screen and the program will halt until you p
ress CONT."
100  PRINT "This will give you time to digest the information on the s
creen at "
110  PRINT "your leisure.)"
120  INPUT "Which of the above choices do you want (1, 2, or 3)?",Choi
ce
130  IF (FRACT(Choice)=0) AND ((Choice>=1) AND (Choice<=3)) THEN 160
140  BEEP
150  GOTO 120
160  IF (Choice=1) OR (Choice=3) THEN LINK "BAR",5000
170  IF (Choice=2) OR (Choice=3) THEN LINK "PIE",6000
180  PRINT PAGE;"   If you want the output of this program dumped from
 the CRT"
190  PRINT "to the internal thermal printer, then answer Y to the ques
tion below."
200  PRINT "If you don't want hardcopy, or your machine doesn't have t
he internal"
210  PRINT "printer, answer N."
220  INPUT "Do you want hardcopy (Y/N)?",A$
230  IF UPC$(A$)="Y" THEN Yhard
240  IF UPC$(A$)="N" THEN Nhard
250  BEEP
260  GOTO 220
270 Yhard:  Hardcopy=1
280  GOTO 300
290 Nhard:  Hardcopy=0
300  PRINT PAGE;
310  PRINT "   First, enter the number of categories in your chart.  T
hen, as the"
320  PRINT "prompt for each category appears, you will enter both the
title of that"
330  PRINT "category, as well as the data value corresponding to it."
340  INPUT "How many categories are there?",N
350  CALL Driver(N,Hardcopy,Choice,Repeat)
360  IF Repeat THEN 20
```

```
370  END
380  SUB Driver(N,Hardcopy,Choice,Repeat)
390  RANDOMIZE
400  OPTION BASE 1
410  DIM A(N),L$(N),Units$[30]
420  Changed=0
430  PRINT
440  FOR I=1 TO N      ! Enter the data
450  DISP "Please enter the title of category #"&VAL$(I);
460  INPUT "",L$(I)
470  DISP "Please enter the value for "&L$(I);
480  INPUT "",A(I)
490  PRINT VAL$(I)&".  "&L$(I)&":  "&VAL$(A(I))
500  NEXT I
510  INPUT "Would you like to make any changes in the above data (Y/N)
?",A$
520  IF UPC$(A$)="N" THEN 730
530  IF UPC$(A$)="Y" THEN 560
540  BEEP
550  GOTO 510
560  PRINT LIN(2),"  Please enter the subscript of the data item you w
ish to change in"
570  PRINT "response to the prompt "&CHR$(34)&"Please enter the subscr
ipt of the item you wish"
580  PRINT "to change"&CHR$(34)&".  If you don't wish to make any chan
ges, press CONT "
590  PRINT "without entering anything.",LIN(2)
600  Changed=1
610  I=PI
620  INPUT "Please enter the subscript of the item you wish to change"
,I
630  IF I=PI THEN 730
640  I=INT(I)
650  IF (I>=1) AND (I<=N) THEN 680
660  BEEP
670  GOTO 610
680  EDIT "Make any changes in the title and press CONT",L$(I)
690  DISP "Enter a new value for "&L$(I)&" or press CONT to keep the o
ld one.";
700  INPUT "",A(I)
710  PRINT VAL$(I)&".  "&L$(I)&":  "&VAL$(A(I))
720  GOTO 610
730  IF NOT Changed THEN 760
740  PRINT LIN(1),"Corrected Data:"
750  GOSUB Print_data
760  INPUT "What units are you using (dollars, hours, people, etc.)?",
Units$
```

```
761  INPUT "Please enter the title of the plot",Title$
770  IF (Choice=1) OR (Choice=3) THEN CALL Bar(A(*),L$(*),N,Units$,Tit
le$)
780  BEEP
790  IF (Choice=1) OR (Choice=3) THEN PAUSE
800  IF ((Choice=1) OR (Choice=3)) AND Hardcopy THEN DUMP GRAPHICS
810  IF (Choice=2) OR (Choice=3) THEN CALL Pie(A(*),L$(*),N,Units$,Tit
le$)
820  BEEP
830  IF (Choice=2) OR (Choice=3) THEN PAUSE
840  IF ((Choice=2) OR (Choice=3)) AND Hardcopy THEN DUMP GRAPHICS
850  EXIT GRAPHICS
860  INPUT "Do you want to run again (Y/N)?",A$
870  IF UPC$(A$)="N" THEN 1010
880  IF UPC$(A$)="Y" THEN 910
890  BEEP
900  GOTO 860
910  INPUT "Do you want to modify the old data, or use new data (O/N)?
",A$
920  IF UPC$(A$)="N" THEN 990
930  IF UPC$(A$)="O" THEN 960
940  BEEP
950  GOTO 910
960  PRINT LIN(1),"Current Data:"
970  GOSUB Print_data
980  GOTO 510
990  Repeat=1
1000 SUBEXIT
1010 Repeat=0
1020 SUBEXIT
1030 Print_data: FOR I=1 TO N
1040             PRINT VAL$(I)&".  "&L$(I)&":  "&VAL$(A(I))
1050             NEXT I
1060             Changed=0
1070             RETURN
```

Subprogram Name: Bar

This subprogram plots bar graphs on the CRT. The graph includes raw numbers, percentages, titles and labels.

Subprogram Utilization:

File Name: BAR
Calling Syntax: CALL Bar (A(*), L$(*), N, Units$, Title$)

Input Parameters:

A(*) - This one-dimensional full-precision numeric array contains the values of each data call

L$(*) - This one-dimensional string array contains the titles of each data cell

N - This full-precision variable, constant, or expression tells how many data cells there are

Units$ - This string holds the units that the number are in (used for titling)

Title$ - This string is a title for the graph

Subprograms Required:

Laxes

Local Variables (Subprogram Bar):

A(*) - Holds the values of each data cell

I - Loop counter

L$(*) - Holds the titles of each data cell

Ly - Log (base 10) of the maximum element (used for computing the distance between tic marks)

Max - The largest data cell

N - The number of data cells

Sum - The sum of the data cells (used for computing percentages)

Testytic - Used for computing tic marks

Title$ - The title of the plot

Units$ - The units for the data cells

Xtic - The distance between tic marks on the X axis

Ytic - The distance between tic marks on the Y axis

Local Variables (Subprogram Laxes):

I - Loop counter

Minticsize - The length (in GDU's) of the minor tic marks

Xfudge - The distance from the Y axis for labels (for clarity of reading)

Xmaj - The major tic count on the X axis

Xmax - The maximum X value on the plot

Xmin - The minimum X value on the plot

Xorg - The X value where the Y axis intercepts the X axis

Xtic - The distance between tic marks on the X axis

Yfudge - The distance from the X axis for labels (for clarity of reading)

Ymaj - The major tic count on the Y axis

Ymax - The maximum Y value on the plot

Ymin - The minimum Y value on the plot

Yorg - The Y value where the X axis intercepts the Y axis

Ytic - The distance between tic marks on the Y axis

Special Considerations and Programming Hints:

1. If the labels of the data cells run off the bottom of the screen, make them shorter.
2. The provided driver (file "BARPIE") will drive either the bar graphs subprogram, or the pie charts subprogram, or both.

3. System Configuration:

Standard Memory Option
(Optional) Internal Thermal Printer
CRT graphics hardware
Graphics ROM

# Annotated Listing of Subprogram Bar

```
5000 SUB Bar(A(*),L$(*),N,Units$,Title$)
5010 PLOTTER IS "GRAPHICS"
5020 GCLEAR
5030 GRAPHICS
5040 PEN 1
5050 Sum=0
5060 Max=A(1)
5070 FOR I=1 TO N
5080 Sum=Sum+A(I)
5090 Max=MAX(Max,A(I))
5100 NEXT I
5110 Ly=LGT(Max)
5120 SETGU
5130 LOCATE 25,120,40,95      ! Set screen boundaries and scale
5140 CLIP 0,123,40,100
5150 MOVE 61.5,97
5160 LORG 5
5170 LABEL USING 5180;Title$
5180 IMAGE #,K
5190 SCALE 0,N+.5,0,Max+Max/10
5200 Xtic=0     ! No tic marks on horizontal axis
5210 Testytic=FRACT(Ly)+(Ly<0)
5220 Ytic=10^(INT(Ly)-1)*(1+1.5*((Testytic>.39794) AND (Testytic<.6989
7))+4*((Testytic>=.69897) AND (Testytic<=.87506))+6.5*(Testytic>.87506
))
5230            ! Compute the vertical tic marks
5240 CALL Laxes(Xtic,Ytic,0,0,1,1,2,0,N+.5,0,Max+Max/10)
5250 LORG 4
5260 FIXED 1
5270 FOR I=1 TO N
5280 MOVE I-.5,0
5290 IPLOT 0,A(I),-1     ! Draw the bar for the ith data item
5300 IPLOT 1,0
5310 IPLOT 0,-A(I)
5320 MOVE I,A(I)
5330 LABEL USING 5340;VAL$(A(I)/Sum*100)&"%"   ! Label the percentages
5340 IMAGE   K
5350 NEXT I
5360 DEG
5370 LDIR 90          ! Rotate the label 90 degrees
5380 SETGU
5390 LORG 5
5400 MOVE 5,70
5410 LABEL USING 5340;Units$ ! Write the display units being used alon
g the
5420                          !  the vertical axis
5430 CLIP 0,123,0,100
```

```
5440 SETUU
5450 FOR I=1 TO N
5460 MOVE I,-Max/20
5470 LORG 8
5480 PEN 1
5490 LDIR 90                          ! Rotate the labels 90 degrees
5500 LABEL USING 5340;L$(I)
5510 NEXT I
5520 BEEP
5530 SUBEXIT
5540 SUB Laxes(Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize,Xmin,Xmax,Ymin
,Ymax)
5550 IF (Xmin>=Xmax) OR (Ymin>=Ymax) THEN SUBEXIT
5560 Xfudge=.02*(Xmax-Xmin)
5570 Yfudge=.02*(Ymax-Ymin)
5580 AXES Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize
5590 DEG
5600 IF NOT Xtic THEN Labely
5610 Labelx:IF SGN(Xtic)=-1 THEN Parx
5620 LDIR 90
5630 LORG 8
5640 GOTO 5670
5650 Parx: LDIR 0
5660 LORG 6
5670 FOR I=Xorg TO Xmax STEP ABS(Xtic)
5680 MOVE I,Yorg-Yfudge
5690 LABEL USING 5700;I
5700 IMAGE #,K
5710 NEXT I
5720 Labely: IF NOT Ytic THEN SUBEXIT
5730 IF SGN(Ytic)=-1 THEN Pary
5740 LDIR 0
5750 LORG 8
5760 GOTO 5790
5770 Pary: LDIR -90
5780 LORG 6
5790 FOR I=Yorg TO Ymax STEP ABS(Ytic)
5800 MOVE Xorg-Xfudge,I
5810 LABEL USING 5700;I
5820 NEXT I
5830 SUBEXIT
```

Driver Utilization:

File Name: BARPIE

This driver allows the user to enter the titles and values of as many data cells as he wants. He may edit the values and titles at any time. Then this data is displayed graphically in the form of either a bar graph or a pie chart, or both. In addition, the graphs may be dumped to the internal thermal printer for hardcopy if the user desires. (The dump is optional - it is not necessary to have the internal printer to run this program.)

User Instructions:

1. Insert the Utility Library cartridge 1 into the primary transport (i.e., the transport above the special function keys).
2. Load the program:
   a. Type: LOAD "BARPIE:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Which of the above choices do you want (1, 2, or 3)?" appears in the display area of the CRT:
   a. Enter: 1, 2, or 3, depending upon whether you want to use the Bar Graph, the Pie Chart, or both, respectively.
   b. Press: CONT
5. When "Do you want hardcopy (Y/N)?" appears in the display area:
   a. If you want to use the CRT/internal thermal printer dump feature:
      1) Type: Y
      2) Press: CONT

      or

   a. If you do not want to use the CRT/internal thermal printer dump feature:
      1) Type: N
      2) Press: CONT

6. When "How many categories are there?" appears in the display area of the CRT:
   a. Enter: The number of categories, or data cells, that you want to use.
   b. Press: CONT
7. Repeat this step as often as necessary.
   a. When "Please enter the title of category #i" appears in the display area:
      1) Type: The title of the ith category.
      2) Press: CONT
   b. When "Please enter the value for [*title*]" appears in the display area:
      1) Enter: The numeric value to be associated with the indicated category.
      2) Press: CONT
   c. Repeat step 7 as often as necessary.
8. When "Would you like to make any changes in the above data (Y/N)?" appears in the display area of the CRT:
   a. If you want to make any changes:
      1) Type: Y
      2) Press: CONT
      3) Go to step 9.

      or
   a. If you do not want to make any changes:
      1) Type: N
      2) Press: CONT
      3) Go to step 12.
9. When "Please enter the subscript of the item you wish to change" appears in the display:
   a. If you want to make a change:
      1) Enter: The subscript of the item you wish to change.
      2) Press: CONT
      3) Go to step 10.

      or
   a. If you do not wish to make any changes:
      1) Press: CONT
      2) Go to step 12.

10. When "Make any changes in the title and press CONT" appears in the display area:
    a. Make any desired changes in the title.
    b. Press: CONT
11. When "Enter a new value for [*title*] or press CONT to keep the old one" appears in the display area:
    a. If you wish to change the old value of the indicated category:
        1) Enter: The new value.
        2) Press: CONT
        3) Go to step 9.

        or

    a. If you do not wish to change the old value of the indicated category:
        1) Press: CONT
        2) Go to step 9.
12. When "What units are you using (dollars, hours, people, etc.)?" appears in the display area:
    a. Type: The units you are using.
    b. Press: CONT
13. When "Please enter the title of the plot" appears in the display area:
    a. Type: The title of the plot.
    b. Press: CONT
14. If you asked for a bar graph in step 4, the bar graph will be drawn, and a beep will sound. The graph will remain on the screen for as long as you want.
    a. To proceed with the program, press CONT.
15. If you asked for a pie chart in step 4, the pie chart will be drawn, and a pair of crosshairs will appear in the center of the pie in the upper left corner.
    a. Read the bottom line in the upper right hand quadrant of the CRT.
    b. Using the DISPLAY keys on the 9845 keyboard, position the crosshairs in the upper left pie to where you would like the value of the category described to be labelled.

c. Press: CONT

d. Now the crosshairs will appear in the center of the lower right pie. Using the DISPLAY keys, position the crosshairs in the lower right pie to where you would like the percentage value of the category described in part 15a to be labelled.

e. Press: CONT

f. Repeat steps 15a-15e as often as necessary.

g. A beep will sound upon completion of the pie chart. When you are ready for the program to proceed, press CONT.

16. When "Do you want to run again (Y/N)?" appears in the display area of the CRT:

a. If you want to run the program again:

1) Type: Y

2) Press: CONT

3) Go to step 17.

or

a. If you do not want to run the program again:

1) Type: N

2) Press: CONT

3) The program ends.

17. When "Do you want to modify the old data, or use new data (O/N)?" appears in the display area:

a. If you want to use the old data:

1) Type: O

2) Press: CONT

3) Go to step 8.

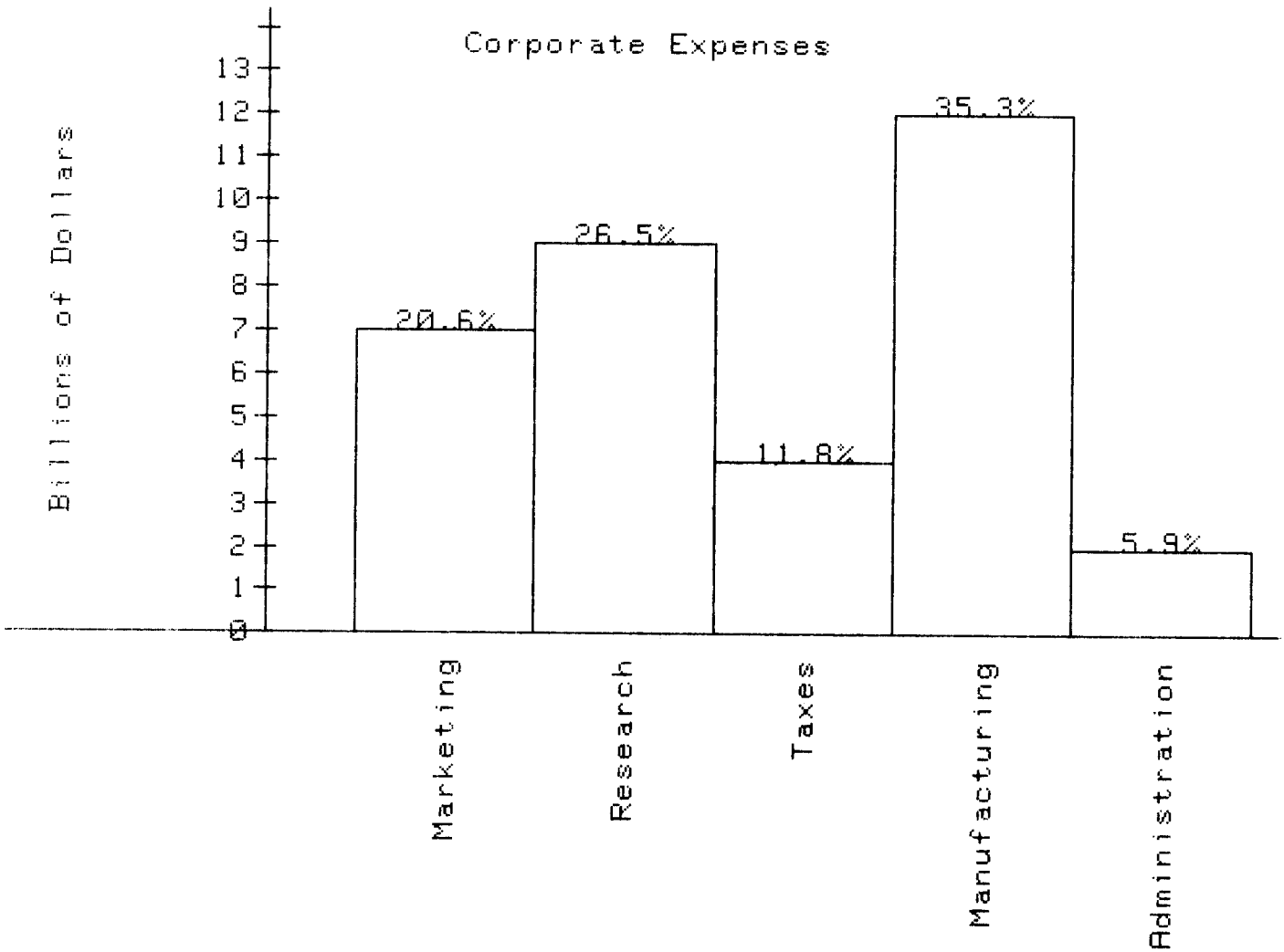
or

a. If you want to use new data:

1) Type: N

2) Press: CONT

3) Go to step 4.


Corporate Expenses
Billions of Dollars
13
12
11
10
9
8
7
6
5
4
3
2
1
0
20.6%
26.5%
11.8%
35.3%
5.9%
Marketing
Research
Taxes
Manufacturing
Administration

## Driver Listing

```
10   PRINTER IS 16
20   PRINT PAGE;"   This program allows the user to enter some data an
d have a bar chart,"
30   PRINT "a pie chart, or both, computed and drawn.  Below you are a
sked to indicate"
40   PRINT "which choice you prefer.",LIN(1)
50   PRINT SPA(10);"1 -- Bar Chart"
60   PRINT SPA(10);"2 -- Pie Chart"
70   PRINT SPA(10);"3 -- Both of the above"
80   PRINT LIN(1);"(After each chart is drawn, a beep will sound.  The
 chart will"
90   PRINT "remain on the screen and the program will halt until you p
ress CONT."
100  PRINT "This will give you time to digest the information on the s
creen at "
110  PRINT "your leisure.)"
120  INPUT "Which of the above choices do you want (1, 2, or 3)?",Choi
ce
130  IF (FRACT(Choice)=0) AND ((Choice)=1) AND (Choice(=3)) THEN 160
140  BEEP
150  GOTO 120
160  IF (Choice=1) OR (Choice=3) THEN LINK "BAR",5000
170  IF (Choice=2) OR (Choice=3) THEN LINK "PIE",6000
180  PRINT PAGE;"   If you want the output of this program dumped from
 the CRT"
190  PRINT "to the internal thermal printer, then answer Y to the ques
tion below."
200  PRINT "If you don't want hardcopy, or your machine doesn't have t
he internal"
210  PRINT "printer, answer N."
220  INPUT "Do you want hardcopy (Y/N)?",A$
230  IF UPC$(A$)="Y" THEN Yhard
240  IF UPC$(A$)="N" THEN Nhard
250  BEEP
260  GOTO 220
270 Yhard:  Hardcopy=1
280  GOTO 300
290 Nhard:  Hardcopy=0
300  PRINT PAGE;
310  PRINT "   First, enter the number of categories in your chart.  T
hen, as the"
320  PRINT "prompt for each category appears, you will enter both the
title of that"
330  PRINT "category, as well as the data value corresponding to it."
340  INPUT "How many categories are there?",N
350  CALL Driver(N,Hardcopy,Choice,Repeat)
360  IF Repeat THEN 20
```

```
370  END
380  SUB Driver(N,Hardcopy,Choice,Repeat)
390  RANDOMIZE
400  OPTION BASE 1
410  DIM A(N),L$(N),Units$[30]
420  Changed=0
430  PRINT
440  FOR I=1 TO N      ! Enter the data
450  DISP "Please enter the title of category #"&VAL$(I);
460  INPUT "",L$(I)
470  DISP "Please enter the value for "&L$(I);
480  INPUT "",A(I)
490  PRINT VAL$(I)&".  "&L$(I)&":  "&VAL$(A(I))
500  NEXT I
510  INPUT "Would you like to make any changes in the above data (Y/N)
?",A$
520  IF UPC$(A$)="N" THEN 730
530  IF UPC$(A$)="Y" THEN 560
540  BEEP
550  GOTO 510
560  PRINT LIN(2),"  Please enter the subscript of the data item you w
ish to change in"
570  PRINT "response to the prompt "&CHR$(34)&"Please enter the subscr
ipt of the item you wish"
580  PRINT "to change"&CHR$(34)&".  If you don't wish to make any chan
ges, press CONT "
590  PRINT "without entering anything.",LIN(2)
600  Changed=1
610  I=PI
620  INPUT "Please enter the subscript of the item you wish to change"
,I
630  IF I=PI THEN 730
640  I=INT(I)
650  IF (I>=1) AND (I<=N) THEN 680
660  BEEP
670  GOTO 610
680  EDIT "Make any changes in the title and press CONT",L$(I)
690  DISP "Enter a new value for "&L$(I)&" or press CONT to keep the o
ld one.";
700  INPUT "",A(I)
710  PRINT VAL$(I)&".  "&L$(I)&":  "&VAL$(A(I))
720  GOTO 610
730  IF NOT Changed THEN 760
740  PRINT LIN(1),"Corrected Data:"
750  GOSUB Print_data
760  INPUT "What units are you using (dollars, hours, people, etc.)?",
Units$
```

```
761  INPUT "Please enter the title of the plot",Title$
770  IF (Choice=1) OR (Choice=3) THEN CALL Bar(A(*),L$(*),N,Units$,Tit
le$)
780  BEEP
790  IF (Choice=1) OR (Choice=3) THEN PAUSE
800  IF ((Choice=1) OR (Choice=3)) AND Hardcopy THEN DUMP GRAPHICS
810  IF (Choice=2) OR (Choice=3) THEN CALL Pie(A(*),L$(*),N,Units$,Tit
le$)
820  BEEP
830  IF (Choice=2) OR (Choice=3) THEN PAUSE
840  IF ((Choice=2) OR (Choice=3)) AND Hardcopy THEN DUMP GRAPHICS
850  EXIT GRAPHICS
860  INPUT "Do you want to run again (Y/N)?",A$
870  IF UPC$(A$)="N" THEN 1010
880  IF UPC$(A$)="Y" THEN 910
890  BEEP
900  GOTO 860
910  INPUT "Do you want to modify the old data, or use new data (O/N)?
",A$
920  IF UPC$(A$)="N" THEN 990
930  IF UPC$(A$)="O" THEN 960
940  BEEP
950  GOTO 910
960  PRINT LIN(1),"Current Data:"
970  GOSUB Print_data
980  GOTO 510
990  Repeat=1
1000 SUBEXIT
1010 Repeat=0
1020 SUBEXIT
1030 Print_data: FOR I=1 TO N
1040             PRINT VAL$(I)&",  "&L$(I)&":  "&VAL$(A(I))
1050             NEXT I
1060             Changed=0
1070             RETURN
```

Subprogram Name: Block

This subprogram plots out figures using relative data stored on a mass memory data file, given an(x,y) coordinate pair as a reference point, and a scale factor telling how large (or small) to make the figure.

A data file containing points digitized from an alphanumeric character set has been provided, so the user may construct overhead slides.

Subprogram Utilization:

File Name: BLOCKS
Calling Syntax: CALL Block (#2, Scale_factor, X,Y,M$,A$(*), A(*))

Input Parameters:

#2 - The file number of the file containing the data points. The file must be assigned in the user's calling program

Scale_factor - This full precision variable, constant, or expression is multiplied with each data point and determines the size of the plotted figure.

X - The X coordinate of the starting point

Y - The Y coordinate of the starting point
(Note: Since the user is allowed to select his own (x,y) starting point, he is also responsible for establishing the SCALE of the picture.)

M$ - This string may be any length, and it contains character codes that tell which figure in data file #2 (for more detail, refer to the section under Methods and Formulae).

A$(*) - This one-dimensional string array is dimensioned in the user's calling program, and contains character codes for each available figure in data file #2 (for more detail, refer to the section under Methods and Formulae).

A(*) - This full precision array is dimensioned in the user's calling program. It is one-dimensional, and each element indicates the number of points that make up the figure defined by the corresponding element in A$(*).

Local Variables:

A - Temporary variable for relative X coordinate

A$ - Temporary variable for the character code of each figure to be plotted

A$(*) - This array contains the character codes of each available figure

A(*) - Each element of this array tells how many points the corresponding figure in A$(*) has

B - Temporary variable for relative Y coordinate

C - Pen code for each data point (A, B)

Count - Loop counter

I - Loop counter

J - Loop counter

M$ - Holds the character codes of the figures to be plotted

Nochars - The number of characters in M$

Scale_factor- The size of the figure

Startpoint - Leftmost starting point of M$ (figures in M$ are centered from left to right about (x,y)

Sum - The starting record number in the data file of the plotted figure

X - X coordinate of starting point

Y - Y coordinate of starting point

Special Considerations and Programming Hints:

1. The character set of the provided data file is as follows:

A - Z
a - z
Ø - 9

!$%&( ) - /: ; . , ? ' ¢ “ ”

The character codes in A$(*) (see the section under Input Parameters) correspond to the above characters with 3 exceptions:

The character code for ¢ is @
The character code for “ is #
The character code for ” is \

The exceptions had to be made because 1) The 9845A keyboard has no ¢ character, and 2) and 3) no distinction is made on the 9845A keyboard between opening and closing quotes.

2. The user may provide his own data if he wants a different style of block lettering, or if he want to use the subprogram to plot figures other than alphanumeric characters. To do so, please study the section under Methods and Formulae.
3. System Configuration:

Standard memory option
CRT graphics hardware
(Optional) - other plotting device
Graphics ROM

## Annotated Listing of Subprogram Block

```
550  SUB Block(#2,Scale_factor,X,Y,M$,A$(*),A(*))
560  IF Scale_factor>0 THEN 600    ! Check for positive scale factor
570  BEEP
580  DISP "ILLEGAL SCALE FACTOR"
590  SUBEXIT
600  Nochars=LEN(M$)               ! Find the number of characters in t
he message
610  Startpoint=-(Nochars-1)*Scale_factor/2  ! Find the starting point
.
620  FOR Count=1 TO Nochars        ! Search directory array for each ch
aracter
630  A$=M$[Count,Count]            ! If the character is not found, the
n leave
640  FOR I=1 TO 79                 !  a blank space and look for the ne
xt one.
650  IF A$=A$(I) THEN Found        ! If charactr is found, then draw it
.
660  NEXT I
670  GOTO 860
680 Found:  Sum=0
690  FOR J=1 TO I-1                ! Each element of the array A(*) has
 the
700  Sum=Sum+A(J)                  !  number of points it takes to draw
 the
710                                !  corresponding character in A$(*).
  By
720                                !  summing up the number of points f
alling
730                                !  before the character, the startin
g position
740                                !  of the character in the point fil
e (#2) is
750                                !  found
760  NEXT J
770  READ #2,Sum+1                 !  Position the serial pointer.
780  GRAPHICS
790  MOVE Startpoint+Scale_factor*(Count-1)+X,Y ! Move to the proper s
tarting
800                                              !  place for each char
acter.
810  FOR J=1 TO A(I)
820  READ #2;A,B,C                 ! Read in the points, one by one, an
d draw
830                                !  the character.
840  IPLOT Scale_factor*A,Scale_factor*B,C
850  NEXT J
860  NEXT Count                    ! Get the next character
870  SUBEXIT
```

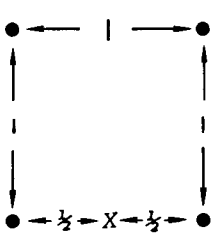
Methods and Formulae:

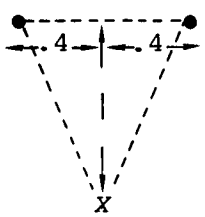
The data file which holds the points to be plotted should have a defined record size of 24 bytes. Each defined record needs to hold three full-precision numbers. (Each number takes 8 bytes; hence the 24 byte defined record size.) The first number is the distance moved in the horizontal direction, the second number is the distance moved in the vertical direction, and the third number is a pen code, telling whether the pen should be up or down during the move. (This determines whether or not a line is drawn during the move.) The valid pen codes follow:

If the pen code is odd, the pen will be lowered.
If the pen code is even, the pen will be raised.

If the sign of the pen code is negative the pen will be raised or lowered before the pen is moved.
If the sign of the pen code is positive, the pen will be raised or lowered after the pen is moved.

Two vectors are necessary for the subprogram Block to interpret the file of data points correctly. One, a string vector (A$(*)), contains the character codes. The other vector, a full precision numeric vector (A(*)), tells the number of points in each figure. For example, suppose your data file contains the points for a square and a triangle. A possible geometry for these two figures is given below. In each, "X" is assumed to be the starting point.



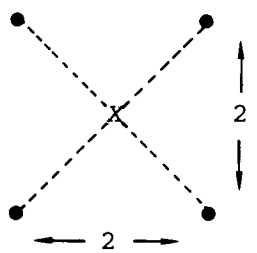
For the above figures, the data file would be as follows:

```
Record #1        .5       0       -1
Record #2         0       1       -1
Record #3        -1       0       -1
Record #4         0      -1       -1
Record #5        .5       0       -1
Record #6        .4       1       -1
Record #7       -.8       0       -1
Record #8        .4      -1       -1
```

Now, assuming that the square's character code will be "S", and the triangle's character code will be "T", the arrays A$(*) and A(*) will be as follows:

```
A$(1) = "S",    A(1) = 5
A$(2) = "T",    A(2) = 3
```

If a third figure, an "X", whose geometry is shown below, were to be added, the arrays and data file would have a different state:




Data File:

```
Record #1        .5       0       -1
Record #2         0       1       -1
Record #3        -1       0       -1
Record #4         0      -1       -1
Record #5        .5       0       -1
Record #6        .4       1       -1
Record #7       -.8       0       -1
```

| | | | |
|---|---|---|---|
| Record #8 | .4 | -1 | -1 |
| Record #9 | 1 | 1 | -2 |
| Record #10 | -2 | -2 | -1 |
| Record #11 | 0 | 2 | -2 |
| Record #12 | 2 | -2 | -1 |

```
A$(1) = "S"  ,  A(1) = 5
A$(2) = "T"  ,  A(2) = 3
A$(3) = "X"  ,  A(3) = 4
```

The starting record number of the figure corresponding to the character code in A$(i) can be found using the following formula:

$$\text{Record number} = \left[\sum_{j=1}^{i-1} A(i)\right] + 1$$

Driver Utilization:

File Name: BLOCKD

The provided driver calls the subprogram Block for use with the data file "ALPHA" which contains 2565 data points. These data points define a 79 member character set. The file "ALPHAD" contains the arrays A$(*) and A(*). (For clarification, see the sections under Methods and Formulae, Input Parameters, and Special Considerations and Programming Hints.)

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary transport (i.e., the transport above the special function keys).
2. Load the program:
   a. Type: LOAD "BLOCKD: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "ENTER MESSAGE" appears in the display area:
   a. Type: A line of text which you want to appear as a single line of blocked letters on the CRT.
   b. Press: CONT
5. When "Now enter the size of the characters (from 1 to 40, 1 is the smallest)" appears in the display area:
   a. Enter: The character size (40 will fill half the CRT, and 1 is 1/40 of that, etc.).
   b. Press: CONT
   c. If "Line is too long for that character size" appears in the display area, then there are too many characters to fit on the screen if they are going to be the specified size. Go back to step 4.
6. When "After reading the above instructions, press CONT to proceed" appears in the display area:
   a. Read the instructions printed on the CRT.
   b. Press: CONT

7. A set of crosshairs will appear on the CRT. Using the keys marked DISPLAY on the 9845A keyboard (left arrow, up arrow, right arrow, down arrow), position the crosshairs to where you want to center the line of text entered in step 4.
   a. Press: CONT
8. Now, if there is room on the screen for all the characters to appear centered about the digitized point, the characters will be plotted. When this process is complete, go to step 4.
9. To halt the program, press STOP.

The quick brown fox

JUMPED

over

the lazy dog

# Driver Listing

```
10   LINK "BLOCKS",550
20   ASSIGN #1 TO "ALPHAD"
30   ASSIGN #2 TO "ALPHA"
40   READ #1;A$(*),A(*)
50   OPTION BASE 1
60   First=1
70   PLOTTER IS "GRAPHICS"
80   DIM A$(79)[1],A(79),M$[30],A$[1]
90   SCALE -40,40,-40,40
100  PRINTER IS 16
110  PRINT PAGE;"This program allows you to have text drawn on the CRT
 in block"
120  PRINT "letters.  You may specify the character size of each line
of text, and its "
130  PRINT "vertical displacement.  The program will handle centering.
  You will"
140  PRINT "be informed if the text line is too long to fit on the scr
een in the"
150  PRINT "specified character size.  The valid character set is give
n below."
160  PRINT LIN(1),"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy
z1234567890!$%&()-/:;.,?'"
170  PRINT
180  PRINT "In addition, a cent sign can be obtained by entering a "&C
HR$(34)&"@"&CHR$(34)&"."
190  PRINT "An opening quotation mark is given by a "&CHR$(34)&"#"&CHR
$(34)&", and a closing quote"
200  PRINT "is given by a "&CHR$(34)&"\"&CHR$(34)&"."
210  PRINT LIN(1),"All other characters will be treated as blanks."
220  EXIT GRAPHICS
230  LINPUT "ENTER MESSAGE",M$
240  INPUT "Now enter the size of the characters (from 1 to 40, 1 is t
he smallest)",Size
250  Size=INT(Size)
260  IF (Size>=1) AND (Size<=40) THEN 290
270  BEEP
280  GOTO 240
290  Nocharsmax=80/Size
300  Nochars=LEN(M$)
310  IF Nocharsmax>=Nochars THEN 360
320  BEEP
330  DISP "Line is too long for that character size"
340  WAIT 1000
350  GOTO 220
360  IF NOT First THEN 410
370  First=0
380  PRINT LIN(1),"Press CONT to get into graphics mode.  Once there,
```

```
position the horizontal"
390  PRINT "crosshair where you want this line of text to occur.  Pres
s CONT "
400  PRINT "again and the program will proceed."
410  DISP "After reading the above instructions, press CONT to proceed
."
420  PAUSE
430  GRAPHICS
440  POINTER 0,0
450  DIGITIZE Xval,Yval
460  Runoff=(40-ABS(Xval))*2/Size
470  IF Runoff>=Nochars THEN 530
480  EXIT GRAPHICS
490  BEEP
500  DISP "The message will run off the surface from that position"
510  WAIT 1000
520  GOTO 220
530  CALL Block(#2,Size,Xval,Yval,M$,A$(*),A(*))
540  GOTO 220
550  END
```

HEWLETT hp PACKARD

# SIMPLE FUNCTION PLOT

This program plots a function which is defined by the user. The user specifies the domain of the plot (minimum x and maximum x) and the program will automatically compute the range of the plot (minimum y and maximum y), and will draw and label the axes, as well as plot the function.

Program Utilization:

File Name: FPLOT

Subprograms required:

Laxes - This subprogram draws and labels the axes in the plot

Variables (Main program):

I - Loop counter

Inc - X increment (distance between plotted points in the X direction)

Lx - The log(base 10) of the domain of the plot (used to compute the distance between tic marks on the X axis)

Ly - The log (base 10) of the range of the plot (used to compute the distance between tic marks on the Y axis)

Maxx - The maximum x value (user input)

Maxy - The maximum y value (computed)

Minx - The minimum x value (user input)

Miny - The minimum y value (computed)

Nmax - The number of computed points

Start - The starting point of the iterations (Minx-Inc)

Title$ - Holds the title of the plot

X - Local variable for function being plotted (FNX(X))

X(*) - The array holding the X coorinates of the points to be plotted

Xfudge - 20% of the plotting area in the X direction (allows room for labels)

Xtic - The distance between tic marks on the X axis

Y(*) - The array holding the Y coordinates of the points to be plotted

Yfudge - 20% of the plotting area in the Y direction (allows room for labels)

Ytic - The distance between tic marks on the Y axis

Variables (Subprogram Laxes):

I - Loop counter

Minticsize - The length (in GDU's) of the minor tic marks

Xfudge - The distance from the Y axis that labels are put (for clarity of reading)

Xmaj - The major tic count on the X axis

Xmax - The maximum X value on the plot

Xmin - The minimum X value on the plot

Xorg - The X value where the Y axis intercepts the X axis

Xtic - The distance between tic marks on the X axis

Yfudge - The distance from the X axis that labels are put (for clarity of reading)

Ymaj - The major tic count on the Y axis

Ymax - The maximum Y value on the plot

Ymin - The minimum Y value on the plot

Yorg - The Y value where the X axis intercepts the Y axis

Ytic - The distance between tic marks on the Y axis

Special Considerations and Programming Hints:

1. The user should be familiar with the syntax scheme of the 9845A when defining functions to be plotted. If the user tries to STORE an invalid line, the 9845A will issue an error message. (Refer to the Operating and Programming Manual for details).
2. Repeated use of this program may make it attractive for the user to define a special function key as a typing aid, instead of repeatedly typing "4ØØØ DEF FNX(X) = ". To do this, follow these instructions:
   a. Type: EDIT
   b. Press: Special function key Ø (or whichever key you decide to use)
   c. Press: CLEAR LINE
   d. Type: 4ØØØ DEF FNX(X) =
   e. Press: Special function key Ø (or whichever key you pressed in part b.

   Now, whenever key Ø is pressed, "4ØØØ DEF FNX(X) = " will be typed.
3. If the 9845A is in graphics mode, any keystroke will cause it to change to alphanumeric mode. To get back into graphics mode:
   a. Type: GRAPHICS
   b. Press: EXECUTE

   The user may find it convenient to define a special function key as an immediate execute so that he can switch into graphics mode with a single keystroke. To do this, follow these instructions:
   a. Type: EDIT
   b. Press: Special function key 1 (or whichever key you decide to use)
   c. Press: CLEAR LINE
   d. Type: GRAPHICS
   e. Press: EXECUTE
   f. Press: Special Function key 1 (or whichever key you pressed in part b).

Now, whenever key 1 is pressed, the 9845 will switch into graphics mode.

4. Presently, this program computes the user specified curve for 102 data points. To change this number, the user must change lines 20 and 30 to reflect the desired number of points (see the Operating and Programming manual for details on program editing).

5. System configuration:
   Standard Memory Option
   CRT Graphics Hardware
   Graphics ROM
   (Optional) Internal Thermal Printer (for CRT graphics dumps)

# Listing of Function Plot Program

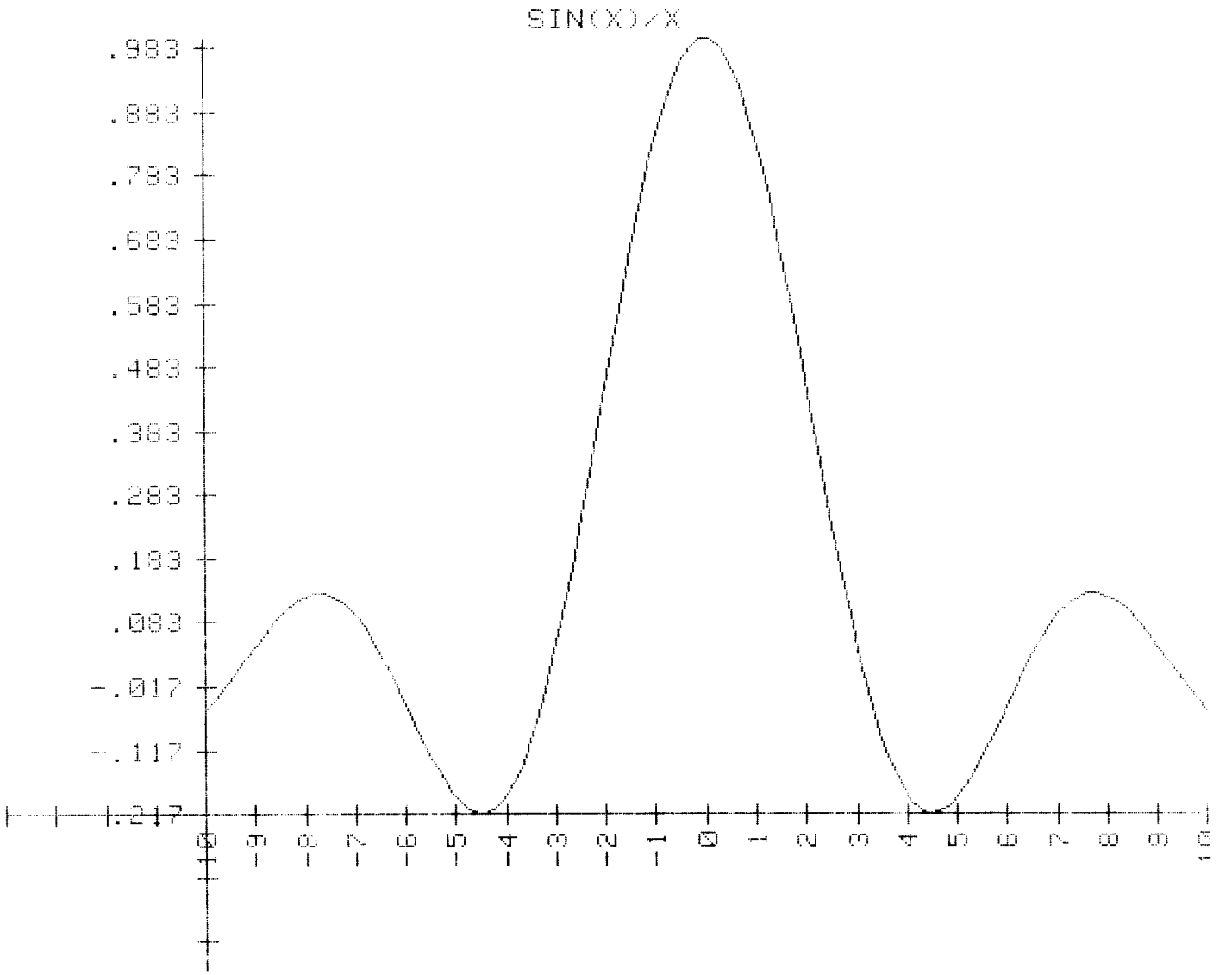
```
10    OPTION BASE 1
20    DIM X(102),Y(102)
30    Nmax=102
40    PLOTTER IS "GRAPHICS"
50    EXIT GRAPHICS
60    DISP "PRESS CONTINUE WHEN YOU ARE READY TO PROCEED "
70    PRINT PAGE;"This is a general function plotter.  You define your
function at"
80    PRINT "line 4000 by typing 4000 DEF FNX(X)=F(X), where F(X) is so
me function"
90    PRINT "of the variable X (i.e. SIN(X), 4*X^3-5*X^2-6, etc.) and p
ressing STORE.",LIN(1)
100   PRINT "After you have done this, press CONT.  If you press CONT w
ithout typing"
110   PRINT "in line 4000, the program will plot SIN(X)/X."
120   PAUSE
130   PRINT PAGE;"Now the program is asking you to determine the range
of the"
140   PRINT "plot.  First enter the minimum X value you want on the gra
ph and press"
150   PRINT "CONT.  Then enter the maximum X value.  The program will c
ompute the"
160   PRINT "minimum and maximum Y values."
170   INPUT "Please enter the minimum X value",Minx
180   PRINT LIN(2),"Minimum X="&VAL$(Minx)
190   INPUT "Please enter the maximum X value",Maxx
200   PRINT "Maximum X="&VAL$(Maxx)
210   IF Minx<Maxx THEN Okay
220   BEEP
230   DISP "INVALID RANGE.  THE MAXIMUM X MUST BE GREATER THAN THE MINI
MUM X."
240   WAIT 1000
250   GOTO 170
260 Okay: INPUT "Please enter the title of the plot",Title$
270       DISP "WORKING..."
280       Miny=9.99999999999E99
290       Maxy=-9.99999999999E99
300       Inc=(Maxx-Minx)/(Nmax-1)
310       Start=Minx-Inc
320       ON ERROR GOTO Bug
330       FOR I=1 TO Nmax
340           X(I)=Start+I*Inc
350           Y(I)=FNX(X(I))
360           IF Y(I)<Miny THEN Miny=Y(I)
370           IF Y(I)>Maxy THEN Maxy=Y(I)
380           GOTO 480
390 Bug:      IF ERRN=31 THEN L31
```

```
400             IF ERRN=29 THEN L29
410             BEEP
420             PRINT ERRM$
430             PAUSE
440             GOTO 480
450 L31:    Y(I)=9.9999999999E99
460         GOTO 480
470 L29:    Y(I)=-9.9999999999E99
480         NEXT I
490         Lx=LGT(Maxx-Minx)
500         Ly=LGT(Maxy-Miny)
510         Xfudge=.2*(Maxx-Minx)
520         Yfudge=.2*(Maxy-Miny)
530         Testxtic=FRACT(Lx)+(Lx<0)
540         Testytic=FRACT(Ly)+(Ly<0)
550         Xtic=10^(INT(Lx)-1)*(1+1.5*((Testxtic>.39794) AND (Testxtic<
.69897))+4*((Testxtic)=.69897) AND (Testxtic<=.87506))+6.5*(Testxtic>.
87506))
560         Ytic=10^(INT(Ly)-1)*(1+1.5*((Testytic>.39794) AND (Testytic<
.69897))+4*((Testytic)=.69897) AND (Testytic<=.87506))+6.5*(Testytic>.
87506))
570         LOCATE 0,123,0,95
580         SCALE Minx-Xfudge,Maxx,Miny-Yfudge,Maxy
590         GRAPHICS
600         CALL Laxes(Xtic,Ytic,PROUND(Minx,-3),PROUND(Miny,-3),1,1,2,M
inx-Xfudge,Maxx,Miny-Yfudge,Maxy)
610         MOVE X(1),Y(1)
620         FOR I=2 TO Nmax
630         DRAW X(I),Y(I)
640         NEXT I
650         SETGU
660         CLIP 0,123,0,100
670         MOVE 61.5,97
680         LORG 5
690         LABEL USING 700;Title$
700         IMAGE #,K
710         BEEP
720         DISP "PROGRAM TERMINATED"
730         END
4000 DEF FNX(X)=SIN(X)/X
5000 SUB Laxes(Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize,Xmin,Xmax,Ymin
,Ymax)
5010 IF (Xmin)=Xmax) OR (Ymin)=Ymax) THEN SUBEXIT
5020 Xfudge=.02*(Xmax-Xmin)
5030 Yfudge=.02*(Ymax-Ymin)
5040 AXES Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize
5050 DEG
```

```
5051 IF NOT Xtic THEN Labely
5060 Labelx:IF SGN(Xtic)=-1 THEN Parx
5070 LDIR 90
5080 LORG 8
5090 GOTO 5120
5100 Parx: LDIR 0
5110 LORG 6
5120 FOR I=Xorg TO Xmax STEP ABS(Xtic)
5130 MOVE I,Yorg-Yfudge
5140 LABEL USING 5150;I
5150 IMAGE #,K
5160 NEXT I
5210 Labely: IF NOT Ytic THEN SUBEXIT
5215 IF SGN(Ytic)=-1 THEN Pary
5220 LDIR 0
5230 LORG 8
5240 GOTO 5270
5250 Pary: LDIR -90
5260 LORG 6
5270 FOR I=Yorg TO Ymax STEP ABS(Ytic)
5280 MOVE Xorg-Xfudge,I
5290 LABEL USING 5150;I
5300 NEXT I
5350 SUBEXIT
```

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary tape transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "FPLOT:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When the message instructing the user to define the function he wants plotted has been printed:
   a. Type: 4ØØØ DEF FNX(X) = f(X), where f(X) is some function of the variable X
   b. Press: STORE
   c. Press: CONT (to resume program execution)
5. When "Please enter the minimum X value" appears in the display area:
   a. Enter: The minimum x value
   b. Press: CONT
6. When "Please enter the maximum X value" appears in the display area:
   a. Enter: The maximum x value
   b. Press: CONT
7. When "Please enter the title of the plot" appears in the display area:
   a. Type: The title of the plot
   b. Press: CONT
8. The graph will be drawn, labelled, and titled.


SIN(X)/X
.983
.883
.783
.683
.583
.483
.383
.283
.183
.083
-.017
-.117
-.217
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10

HEWLETT hp PACKARD

# ITERATIVE PARAMETER PLOT

This program plots a function which is specified by the user. The user specifies the domain of the plot (minimum and maximum X) and the program will automatically compute the range of the plot (minimum and maximum Y) and will draw and label the axes, as well as plot the function.

In addition, the user may modify some parameter(s) of the plotted function, and superimpose the modified function on the original one, thus giving instant visual feedback as to how the modified parameter affects the function.

Program Utilization:

File Name: ITPLOT

Subprograms Required:

Laxes

Variables (Main Program):

I - Loop counter

Inc - X increment (distance between plotted points in the X direction)

Lx - The log (base 10) of the domain of the plot (used to compute the distance between tic marks on the X axis)

Ly - The log (base 10) of the range of the plot (used to compute the distance between tic marks on the Y axis)

Maxx - The maximum X value (user input)

Maxy - The maximum Y value (computed)

Minx - The minimum X value (user input)

Miny - The minimum Y value ( computed)

Start - The starting point of the iterations (Minx-Inc)

X - Local variable for the function being plotted (FNX(X))

X(*) - The array holding the X coordinates of the points to be plotted

Xfudge - 20% of the plotting area in the X direction (allows room for labels)

Xtic - The distance between tic marks on the X axis

Y(*) - The array holding the Y coordinates of the points to be plotted

Yfudge - 20% of the plotting area in the Y direction (allows room for labels)

Ytic - The distance between tic marks on the Y axis

Variables (Subprogram Laxes):

I - Loop counter

Minticsize - The length (in GDU's) of the minor tic marks

Xfudge - The distance from the Y axis that labels are put (for clarity of reading)

Xmaj - The major tic count on the X axis

Xmax - The maximum X value on the plot

Xmin - The minimum X value on the plot

Xorg - The X value where the Y axis intercepts the X axis

Xtic - The distance between tic marks on the X axis

Yfudge - The distance from the X axis that labels are put (for clarity of reading)

Ymaj - The major tic count on the Y axis

Ymax - The maximum Y value on the plot

Ymin - The minimum Y value on the plot

Yorg - The Y value where the X axis intercepts the Y axis

Ytic - The distance between tic marks on the Y axis

Special Considerations and Programming Hints:

1. The user should be familiar with the syntax scheme of the 9845 when defining functions to be plotted. If the user tries to STORE an invalid line, the 9845 will issue an error message. (Refer to the Operating and Programming Manual for details).
2. Repeated use of this program may make it attractive for the user to define a special function key as a typing aid, instead of repeatedly typing "4ØØØ DEF FNX(X) =". To do this, follow these instructions:
   a. Type: EDIT
   b. Press: Special function key Ø (or whichever key you decide to use)
   c. Press: CLEAR LINE
   d. Type: 4ØØØ DEF FNX(X) =
   e. Press: Special function key Ø (or whichever key you pressed in part b.

   Now, whenever key Ø is pressed, "4000 DEF FNX(x)=" will be typed.
3. If the 9845 is in graphics mode, any keystroke will cause it to change to alphanumeric mode. To get back into graphics mode:
   a. Type: GRAPHICS
   b. Press: EXECUTE

   The user may find it convenient to define a special function key as an immediate execute so that he can switch into graphics mode with a single keystroke. To do this, follow these instructions:
   a. Type: EDIT
   b. Press: Special function key 1 (or whichever key you decide to use)
   c. Press: CLEAR LINE
   d. Type: GRAPHICS

e. Press: EXECUTE

f. Press: Special function key 1 (or whichever key you pressed in part b)

Now, whenever key 1 is pressed, the 9845 will switch into graphics mode.

4. Presently, this program computes the user specified curve for 102 data points. To change this number, the user must change lines 20 and 30 to reflect the desired number of points (see the Operating and Programming manual for details on program editing).

5. System Configuration:
Standard Memory Option
CRT Graphics Hardware
Graphics ROM
(Optional) Internal Thermal Printer (for CRT graphics dumps)

# Listing of Iterative Parameter Plot Program

```
10    OPTION BASE 1
20    DIM X(102),Y(102)
30    Nmax=102
40    PLOTTER IS "GRAPHICS"
50    EXIT GRAPHICS
60    DISP "PRESS CONTINUE WHEN YOU ARE READY TO PROCEED "
70    PRINT PAGE;"This is an iterative function plotter.  You define yo
ur function at"
80    PRINT "line 4000 by typing 4000 DEF FNX(X)=F(X), where F(X) is so
me function"
90    PRINT "of the variable X (i.e. SIN(X), 4*X^3-5*X^2-6, etc.) and p
ressing STORE."
100   PRINT "Then the program will ask you to define the X domain of th
e plot, and"
110   PRINT "it will compute the Y range.  After drawing the original p
lot, the"
120   PRINT "program will allow you to modify the function (or change i
t completely,"
130   PRINT "for that matter) and re-draw it within the scales computed
 for the"
140   PRINT "previous plot.  This process may be repeated as often as n
ecessary."
150   PRINT "   Press CONT after having defined the function.  If you p
ress CONT"
160   PRINT "without having defined a function at line 4000, the progra
m will use"
170   PRINT "SIN(X)/X."
180   PAUSE
190   PRINT PAGE;"Now the program is asking you to determine the domain
 of the"
200   PRINT "plot.  First enter the minimum X value you want on the gra
ph and press"
210   PRINT "CONT.  Then enter the maximum X value.  The program will c
ompute the"
220   PRINT "minimum and maximum Y values."
230   INPUT "Please enter the minimum X value",Minx
240   PRINT LIN(1),"Minimum X="&VAL$(Minx)
250   INPUT "Please enter the maximum X value",Maxx
260   PRINT "Maximum X="&VAL$(Maxx)
270   IF Minx<Maxx THEN Okay
280   BEEP
290   DISP "INVALID RANGE.  THE MAXIMUM X MUST BE GREATER THAN THE MINI
MUM X."
300   WAIT 1000
310   GOTO 230
320 Okay: INPUT "Please enter the title of the plot",Title$
330       PRINT PAGE
```

```
340         GOSUB Explain
350         DISP "WORKING..."
360         Miny=9.9999999999E99
370         Maxy=-9.9999999999E99
380         Inc=(Maxx-Minx)/(Nmax-1)
390         Start=Minx-Inc
400         ON ERROR GOTO Bug
410         FOR I=1 TO Nmax
420             X(I)=Start+I*Inc
430             Y(I)=FNX(X(I))
440             IF Y(I)<Miny THEN Miny=Y(I)
450             IF Y(I)>Maxy THEN Maxy=Y(I)
460             GOTO 560
470 Bug:        IF ERRN=31 THEN L31
480             IF ERRN=29 THEN L29
490             BEEP
500             PRINT ERRM$
510             PAUSE
520             GOTO 560
530 L31:    Y(I)=9.9999999999E99
540         GOTO 560
550 L29:    Y(I)=-9.9999999999E99
560         NEXT I
570         Lx=LGT(Maxx-Minx)
580         Ly=LGT(Maxy-Miny)
590         Xfudge=.2*(Maxx-Minx)
600         Yfudge=.2*(Maxy-Miny)
610         Testxtic=FRACT(Lx)+(Lx<0)
620         Testytic=FRACT(Ly)+(Ly<0)
630         Xtic=10^(INT(Lx)-1)*(1+1.5*((Testxtic).39794) AND (Testxtic<
.69897))+4*((Testxtic)=.69897) AND (Testxtic<=.87506))+6.5*(Testxtic).
87506))
640         Ytic=10^(INT(Ly)-1)*(1+1.5*((Testytic).39794) AND (Testytic<
.69897))+4*((Testytic)=.69897) AND (Testytic<=.87506))+6.5*(Testytic).
87506))
650         LOCATE 0,123,0,95
660         SCALE Minx-Xfudge,Maxx,Miny-Yfudge,Maxy
670         GRAPHICS
680         CALL Laxes(Xtic,Ytic,PROUND(Minx,-3),PROUND(Miny,-3),1,1,2,M
inx-Xfudge,Maxx,Miny-Yfudge,Maxy)
690         MOVE X(1),Y(1)
700         FOR I=2 TO Nmax
710         DRAW X(I),Y(I)
720         NEXT I
730         SETGU
740         CLIP 0,123,0,100
750         MOVE 61.5,97
```

```
760         LORG 5
770         LABEL USING 780;Title$
780         IMAGE #,K
790         CLIP 0,123,0,95
800         SETUU
810         BEEP
820         PRINT PAGE;
830         GOSUB Explain
840         DISP "WORKING..."
850         GOTO 970
860 Explain: PRINT "Once the function has been plotted, you may superi
mpose a"
870         PRINT "new curve over the old one by redefining the function
.  You may"
880         PRINT "either type line 4000 again, or you may issue the com
mand EDITLINE"
890         PRINT "4000 (EXECUTE), and change the line directly.  After
you have"
900         PRINT "changed the function, press CONT and the program will
 draw the"
910         PRINT "new function within the boundaries of the old graph.
 When you"
920         PRINT "have finished reading these instructions, press CONT
and the "
930         PRINT "program will proceed.  A beep will sound when the plo
t is finished."
940         DISP "Press CONT when you are ready to proceed"
950         PAUSE
960         RETURN
970         ON ERROR GOTO Bug1
980         FOR I=1 TO Nmax
990            X(I)=Start+I*Inc
1000           Y(I)=FNX(X(I))
1010           GOTO 1110
1020 Bug1:     IF ERRN=31 THEN L131
1030           IF (ERRN=28) OR (ERRN=29) THEN L129
1040           BEEP
1050           PRINT ERRM$
1060           PAUSE
1070           GOTO 1110
1080 L131:     Y(I)=9.99999999999E99
1090           GOTO 1110
1100 L129:     Y(I)=-9.99999999999E99
1110        NEXT I
1120        OFF ERROR
1130        MOVE X(1),Y(1)
1140        GRAPHICS
```

```
1150       FOR I=2 TO Nmax
1160       DRAW X(I),Y(I)
1170       NEXT I
1180       PENUP
1190       GOTO 810
4000 DEF FNX(X)=SIN(X)/X
5000 SUB Laxes(Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize,Xmin,Xmax,Ymin
,Ymax)
5010 IF (Xmin>=Xmax) OR (Ymin>=Ymax) THEN SUBEXIT
5020 Xfudge=.02*(Xmax-Xmin)
5030 Yfudge=.02*(Ymax-Ymin)
5040 AXES Xtic,Ytic,Xorg,Yorg,Xmaj,Ymaj,Minticsize
5050 DEG
5051 IF NOT Xtic THEN Labely
5060 Labelx:IF SGN(Xtic)=-1 THEN Parx
5070 LDIR 90
5080 LORG 8
5090 GOTO 5120
5100 Parx: LDIR 0
5110 LORG 6
5120 FOR I=Xorg TO Xmax STEP ABS(Xtic)
5130 MOVE I,Yorg-Yfudge
5140 LABEL USING 5150;I
5150 IMAGE #,K
5160 NEXT I
5210 Labely: IF NOT Ytic THEN SUBEXIT
5215 IF SGN(Ytic)=-1 THEN Pary
5220 LDIR 0
5230 LORG 8
5240 GOTO 5270
5250 Pary: LDIR -90
5260 LORG 6
5270 FOR I=Yorg TO Ymax STEP ABS(Ytic)
5280 MOVE Xorg-Xfudge,I
5290 LABEL USING 5150;I
5300 NEXT I
5350 SUBEXIT
```

User Instructions:

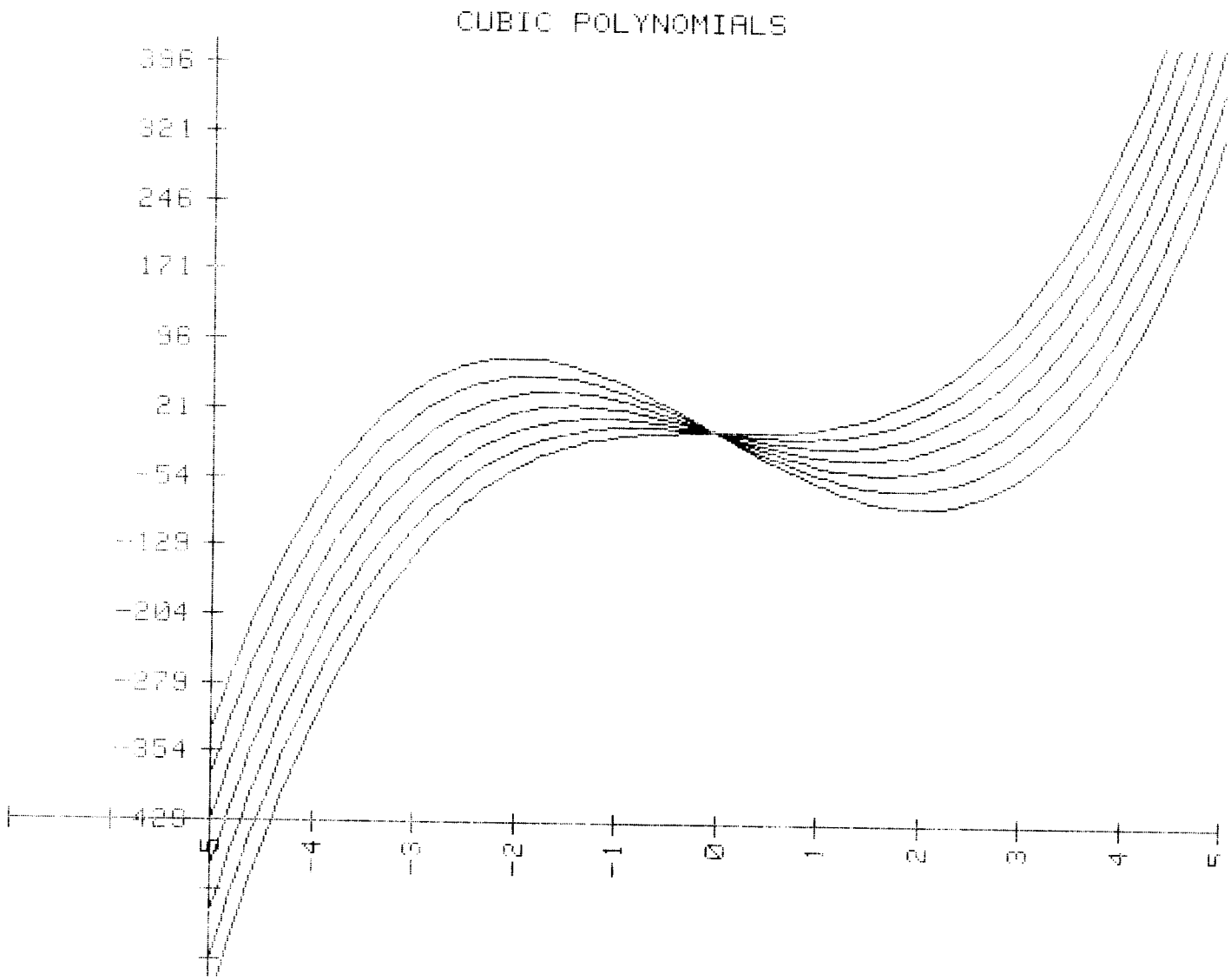
1. Insert the Utility Library cartridge 2 into the primary tape transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "ITPLOT:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When the message instructing the user to define the function he wants plotted has been printed:
   a. Type: 4ØØØ DEF FNX(X)= f(X) where f(X) is some function of the variable X.
   b. Press: STORE
   c. Press: CONT (to resume program execution)
5. When "Please enter the minimum X value" appears in the display area:
   a. Enter: The minimum X value
   b. Press: CONT
6. When "Please enter the maximum X value" appears in the display area:
   a. Enter: The maximum X value
   b. Press: CONT
7. When "Please enter the title of the plot" appears in the display area:
   a. Type: The title of the plot
   b. Press: CONT
8. When "Press CONT when you are ready to proceed" appears in the display area:
   a. Read the instructions printed on the screen as to how to modify the function for subsequent plots.
   b. When you have finished and understand the material on the CRT, press CONT.

9. The function will be plotted and labelled. When the function is finished plotting, a beep will sound. This signifies that the program has paused. At this point, the user has two choices:
   a. Do nothing (the program is finished)
   b. Modify the function and superimpose the modified function over the original plot.
      Note: At any time, the plot may be dumped to the internal thermal printer (if your machine has the internal printer option) via the DUMP GRAPHICS command.

10. If you choose to modify the function, this may be done in two ways:
   a. 1) Type: 4ØØØ DEF FNX(X) = f(X) where f(X) is the new function
      2) Press: STORE
      3) Press: CONT (to resume the program)
      4) Go to step 9.

      or

   b. 1) Type: EDIT LINE 4ØØØ
      2) Press: EXECUTE
      3) Line 4ØØØ will appear in the middle of the screen. The cursor will appear immediately to the right of line 4ØØØ. Make any corrections in line 4ØØØ that you want.
      4) Press: STORE
      5) Press: CONT (to resume program execution)
      6) Go to step 9.


CUBIC POLYNOMIALS
396
321
246
171
96
21
-54
-129
-204
-279
-354
-429
-5
-4
-3
-2
-1
0
1
2
3
4
5

HEWLETT hp PACKARD

# PICTURE CONSTRUCTION AND ENTITY CREATION

This group of programs is used to interactively create pictures on the CRT.

There are three programs in this section. They may be used alone, or in conjunction with each other.

The construction program (file "CONSTR") allows the user to construct line drawings. (These line drawings will hereafter be referred to as "primitives", for reasons which will become apparent later.) These "primitives" may be modified in several ways. Points may be moved, deleted, or inserted, and lines may be inverted (that is, the user may erase a line between two points, or he may make a line that has been erased re-appear). These "primitives" can be stored away on a mass storage file for later use.

The entity reproduction program (file "REPRO") allows the user to use the "primitives" created with the construction program to build more complex entities. Using the special function keys, the user may select a "primitive" from his menu, draw lines, move "primitives" around for fine positioning, delete "primitives", modify his menu (the menu is a selection of "primitives", which the user specifies), load the construction and conversion programs, and save his picture for later use.

The conversion program (file "CONVRT") takes a file that was built using the entity reproduction program and converts it to a "primitive" file which can be used with the entity reproduction program to make even more complex pictures.

HEWLETT hp PACKARD

# PICTURE CONSTRUCTION PROGRAM

This program is used to interactively create line drawings on the CRT. The program features "rubber-banding", that is, a point may be moved around, while still keeping any lines attached to it. Points may also be deleted, or they may be inserted in a line and moved to the position desired (via the "rubber-banding" feature). Pictures can be saved on a mass storage file to be used later on, also.

Program Utilization:

File Name: CONSTR

Subprograms Required:

Lineseek
Seek

Variables (Main program):

| | | |
|---|---|---|
| A | - | Used to hold old X coordinate of the point being moved in case the operation is cancelled |
| A$ | - | Used to hold user's answer to Y/N questions |
| B | - | Used to hold old Y coordinate of the point being moved in case the operation is cancelled |
| C | - | Used with ASSIGN statement to see whether or not a data file is available |
| Entrypoint | - | Used to tell whether the program is being run by itself, or loaded from the entity reproduction program. (1 if stand-alone, 2 if loaded by entity reproduction program) |
| File$ | - | Used to hold the name of the user's data file |
| I | - | Loop counter |
| Increment | - | Set the increment by which the special function keys will move the cursor in the "rubber-band" mode |

J - Loop counter

N - The number of points on the screen

Nmax - The maximum number of points that can be stored in the X(*), Y(*), and P(*) arrays

P(*) - The array containing the pen codes for the points on the screen

Sideinc - Tells how far to the side (left or right) to move the cursor in the "rubber-band" mode

Upinc - Tells how far in the vertical direction (up or down) to move the cursor in the "rubber-band" mode

W - Subscript of the point being affected by the current move, insert, or delete point operation

W1 - Subscript of the first endpoint of the line being affected by the insert point or invert line operation

W2 - Subscript of the second endpoint of the line being affected by the insert or invert line operation

X,Y - Newest point to be drawn or moved to

X(*) - The array containing the X coordinates of the points on the screen

Xmid - The X coordinate of the midpoint of the line affected by the current insert point or invert line operation

Y(*) - The array containing the Y coordinates of the points

Ymid - The Y coordinate of the midpoint of the line affected by the current insert point or invert line operation

Variables (Subprogram Seek):

D - Distance from point (X,Y) to each coordinate in the X(*) and Y(*) arrays

I - Loop counter

Mindis - The minimum distance from (X,Y) to one of the points in the X(*) and Y(*) arrays

N - The number of points in the X(*) and Y(*) arrays

W - The subscript of the point in the X(*) and Y(*) arrays which is the closest to the point (x,y)

X - The X coordinate of the point from which the shortest distance is being computed

X(*) - The array containing the X coordinates of all the points on the screen

Y - The Y coordinate of the point from which the shortest distance is being computed

Y(*) - The array containing the Y coordinates of all the points on the screen

Variables (Subprogram Lineseek):

A - The coefficient of X in the equation of a line Ax + By = C

B - The coefficient of Y in the equation of a line Ax + By = C

D1 - The distance from (X,Y) to the left endpoint of a line segment

D2 - The distance from (X,Y) to the right endpoint of a line segment

Distance - The perpendicular distance from (X,Y) to each line segment on the screen

I - Loop counter

J - Local variable for distance function FND(J)

Mindis - The distance from (X,Y) to the nearest line segment

N - The number of points on the screen

W1 - The first endpoint of the nearest line segment

W2 - The second endpoint of the nearest line segment

X - The X coordinate of the point from which the distances are computed

X(*) - The array holding the X coordinates of all the points

on the screen

Xhat - The X coordinate of the point on a line where a perpendicular dropped from (X,Y) would intersect the line

Xmid - The X coordinate of the midpoint of the nearest line segment

Y - The Y coordinate of the point from which the distances are measured

Y(*) - The array containing the Y coordinates of all the points on the screen

Yhat - The Y coordinate of the point on a line where a perpendicular dropped from (X,Y) would intersect the line

Ymid - The coordinate of the midpoint of the nearest line segment

Special Considerations and Programming Hints:

1. The highest number of points that you can presently draw on the screen is 100. If this number is insufficient, the dimension statement in line 70 should be changed so that X(*), Y(*), and P(*) contain the proper number of points. Also, line 80 must be changed to reflect the new array size.
2. System Configuration:
   Standard Memory Option
   CRT Graphics Hardware
   Graphics ROM
   (Optional) Internal Thermal Printer (for CRT graphics dumps)

## Annotated Program Listing of Picture Construction Program

```
10 Ep1:   Entrypoint=1     ! This program has two entry points.  One is
 for a
20                         !  stand-alone run.
30   GOTO 60
40 Ep2:   Entrypoint=2     ! The other is for a link from the entity re
production
50                         !  program.
60   OPTION BASE 1
70   DIM X(100),Y(100),P(100)
80   Nmax=100              ! Set the maximum number of points the arrays
 will hold
90   MAT X=ZER             ! Initialize the arrays for X and Y coordina
tes and
100  MAT Y=ZER             !  the pen control codes.
110  MAT P=ZER
120  X=Y=.5
130  PLOTTER IS "GRAPHICS"
140  EXIT GRAPHICS
150  LIMIT 35,165,10,140
160  FRAME
170  SCALE 0,1,0,1
180  PEN 1
190  N=0
200  PRINTER IS 16
210  PRINT PAGE;"   This program allows you to build a picture by digi
tization and"
220  PRINT "modify it.  You may either start fresh, or you may build o
n an already"
230  PRINT "existing file."
240  PRINT "   This program will be easier to use if you take a blank
overlay"
250  PRINT "(part no. 7120-6164), and fill in the appropriate squares
as follows:",LIN(1)
260  PRINT "KEY 0:  <-- (left arrow)";TAB(30);"KEY 8:  DRAW"
270  PRINT "KEY 1:   ^  (up arrow)";TAB(30);"KEY 10: MOVE"
280  PRINT "KEY 2:  --> (right arrow)";TAB(30);"KEY 3:  MOVE POINT";TA
B(55);"KEY 11: SET POINT"
290  PRINT "KEY 9:      (down arrow)";TAB(30);"KEY 4:  DELETE POINT"
300  PRINT TAB(30);"KEY 5:  INSERT POINT"
310  PRINT TAB(30);"KEY 13: INVERT LINE"
320  PRINT LIN(1),TAB(45);"KEY 14: SEEK",LIN(1);TAB(20);"KEY 12: CANCE
L";TAB(45);"KEY 6:  ACCEPT";LIN(1),TAB(45);"KEY 7:  REJECT"
330  PRINT LIN(1),"KEY 15: SAVE",LIN(1)
340  INPUT "Do you want to use an old file to begin with (Y/N)?",A$
350  IF UPC$(A$)="Y" THEN Use_old_file
360  IF UPC$(A$)="N" THEN Startfresh
370  BEEP
```

```
380  GOTO 340
390 Use_old_file:  INPUT "Which file do you want to use?",File$
400       ASSIGN #1 TO File$,C
410       IF NOT C THEN 460
420       BEEP
430       DISP File$&" is not available"
440       WAIT 1000
450       GOTO 340
460       READ #1,1;N    ! Read in the number of points in the file.
470       IF N<=Nmax THEN 520
480       BEEP
490       DISP "FILE TOO LARGE"
500       WAIT 1000
510       GOTO 340
520       READ #1,2      ! Set the serial pointer.
530       FOR I=1 TO N   ! Read in the points.
540       READ #1;X(I),Y(I),P(I)
550       NEXT I
560       GOTO Cancel
570 Startfresh:  GOSUB Setkeys1
580 Point:   GRAPHICS       ! Digitize points
590          POINTER X,Y
600          DIGITIZE X,Y
610          GOTO Point
620 Draw:    IF NOT N THEN Move
630          IF N+1<=Nmax THEN 660
640          GOSUB Complain
650          RETURN
660          DRAW X,Y       ! Draw to the digitized point (and save it)
670          P(N+1)=-1
680          GOTO Return
690 Move:    IF N+1<=Nmax THEN 720
700          GOSUB Complain
710          RETURN
720          MOVE X,Y       ! Move to the digitized point (and save it)
730          P(N+1)=-2
740 Return:  N=N+1
750          X(N)=X
760          Y(N)=Y
770          RETURN         ! Go digitize some more
780 Setkeys1:  ON KEY #8 GOSUB Draw       ! Enable the command keys.
790            ON KEY #10 GOSUB Move
800            ON KEY #3 GOTO Movepoint
810            ON KEY #4 GOTO Deletepoint
820            ON KEY #5 GOTO Insertpoint
830            ON KEY #13 GOTO Deleteline
840            ON KEY #15 GOTO Savedata
```

```
850             RETURN
860 Offkeys1:   OFF KEY #8                  ! Disable command keys.
870             OFF KEY #10
880             OFF KEY #3
890             OFF KEY #4
900             OFF KEY #5
910             OFF KEY #13
920             OFF KEY #15
930             RETURN
940 Offkeys_a_r:  OFF KEY #6                ! Disable accept and reject ke
ys.
950               OFF KEY #7
960               RETURN
970 Movepoint:  GOSUB Offkeys1              ! Disable previous keys
980             ON KEY #12 GOTO Cancel      ! Enable cancel and seek keys.
990             ON KEY #14 GOTO Mseek
1000            POINTER X,Y,1
1010            DIGITIZE X,Y
1020            BEEP
1030            GRAPHICS
1040            GOTO 1000
1050 Mseek:   GOSUB Seek                    ! Find the nearest point and draw a
line to it.
1060          ON KEY #6 GOTO Ma             ! Enable accept and reject keys
1070          ON KEY #7 GOTO Mr
1080          GOTO 1080
1090 Ma:      GOSUB Offkeys_a_r             ! Disable accept and reject keys.
1100          GOSUB Undraw
1110          ON KEY #11 GOTO Setpoint
1120          A=X(W)                        ! Save the old points in case of can
cellation
1130          B=Y(W)
1140          Increment=1/45                ! Set the standard increment.
1160          Upinc=Sideinc=0               ! Zero out the Up and Side increment
s.
1170          GOSUB Vkeys                   ! Set vertical cursor control keys.
1180          GOSUB Hkeys                   ! Set horizontal cursor control keys
.
1190          POINTER A,B,2                 ! Use the blinking cursor
1200          GRAPHICS
1210          GOTO 1200                     ! Wait for command
1220 Cursor:  PEN -1                        ! Set pen for erase.
1230          IF W=1 THEN 1300              ! Check to see if the point to be mo
ved is
1240                                        !  the first one.
1250          IF P(W)=-2 THEN 1300          ! Don't erase the previous line if i
t wasn't
```

```
1260                                    !  there to begin with.
1270          MOVE X(W-1),Y(W-1)
1280          DRAW A,B                  ! Erase line before the point
1290          GOTO 1310
1300          MOVE A,B
1310          IF W=N THEN 1350          ! Check to see if the point to be mo
ved is
1320                                    !  the last one.
1330          IF P(W+1)=-2 THEN 1350!Ignore the next line if it doesn't
 exist
1340          DRAW X(W+1),Y(W+1)        ! Erase the line after the point.
1350          A=A+Sideinc               ! Update the new coordinates
1360          B=B+Upinc
1370          PEN 2
1380          IF W=1 THEN 1440          ! Check to see if the point being mo
ved is
1390                                    !  the first one.
1400          IF P(W)=-2 THEN 1440 ! Ignore the previous line if it doe
sn't exist
1410          MOVE X(W-1),Y(W-1)        ! Redraw the line before the point.
1420          DRAW A,B
1430          GOTO 1450
1440          MOVE A,B
1450          IF W=N THEN 1490          ! Check to see if the point being mo
ved is
1460                                    !  the last one.
1470          IF P(W+1)=-2 THEN 1490 ! Ignore the next line if it doesn
't exist
1480          DRAW X(W+1),Y(W+1)        ! Re-draw the line after the point.
1490          Upinc=Sideinc=0           ! Reset the Up and Side increments.
1500          POINTER A,B,2
1510          RETURN
1520 Setpoint: X(W)=A                   ! Store away the present coordinates
1530             Y(W)=B
1540             GOTO Cancel
1550 Mr:  GOSUB Offkeys_a_r             ! Disable the accept and reject keys
.
1560      GOSUB Undraw
1570      GOTO Movepoint
1580 Undraw: PEN -1                     ! Set the pen to erase
1590          MOVE X,Y
1600          DRAW X(W),Y(W)
1610          PEN 1                     ! Reset the pen to draw
1620          MOVE X,Y
1630          RETURN
1640 Complain: BEEP
1650          EXIT GRAPHICS
```

```
1660          DISP "ARRAY OVERFLOW--NO MORE POINTS CAN BE STORED"
1670          WAIT 3000
1680          GRAPHICS
1690          X=X(N)
1700          Y=Y(N)
1710          RETURN
1720 Up: Upinc=Increment
1730     GOTO Cursor
1740 Ups:Upinc=Increment/10
1750     GOTO Cursor
1760 Down:Upinc=-Increment
1770     GOTO Cursor
1780 Downs:Upinc=-Increment/10
1790     GOTO Cursor
1800 Left:Sideinc=-Increment
1810     GOTO Cursor
1820 Lefts:Sideinc=-Increment/10
1830     GOTO Cursor
1840 Right: Sideinc=Increment
1850     GOTO Cursor
1860 Rights:Sideinc=Increment/10
1870     GOTO Cursor
1880 Vkeys:ON KEY #1 GOSUB Up     ! Enable the vertical cursor control
keys
1890     ON KEY #9 GOSUB Down
1900     ON KEY #17 GOSUB Ups
1910     ON KEY #25 GOSUB Downs
1920     RETURN
1930 Hkeys:ON KEY #0 GOSUB Left   ! Enable the horizontal cursor contro
l keys
1940     ON KEY #2 GOSUB Right
1950     ON KEY #16 GOSUB Lefts
1960     ON KEY #18 GOSUB Rights
1970     RETURN
1980 Offkeys2: OFF KEY #0         ! Disable all cursor control keys
1990           OFF KEY #1
2000           OFF KEY #2
2010           OFF KEY #9
2020           OFF KEY #16
2030           OFF KEY #17
2040           OFF KEY #18
2050           OFF KEY #25
2060           RETURN
2070 Offallkeys: GOSUB Offkeys1        ! Turn off all special function
keys
2080             GOSUB Offkeys2
2090             GOSUB Offkeys_a_r
```

```
2100              OFF KEY #11
2110              OFF KEY #12
2120              OFF KEY #13
2130              OFF KEY #14
2140              OFF KEY #15
2150              RETURN
2160 Cancel:    GOSUB Offallkeys    ! Disable all keys
2170            GCLEAR              ! Clear the screen
2180            FRAME
2190            FOR I=1 TO N        ! Redraw what's in the array
2200            PLOT X(I),Y(I),P(I)
2210            NEXT I
2220            GOSUB Setkeys1      ! Enable command keys
2230            PEN 1
2240            X=X(N)              ! Set the cursor coordinates to the
last
2250            Y=Y(N)              !  point on the picture
2260            GOTO Point          ! Wait for the next command
2270 Deletepoint: GOSUB Offkeys1    ! Disable previous keys
2280            ON KEY #12 GOTO Cancel  ! Enable cancel key.
2290            ON KEY #14 GOTO Dseek   ! Enable seek key.
2300            POINTER X,Y
2310            DIGITIZE X,Y
2320            BEEP
2330            GRAPHICS
2340            GOTO 2300
2350 Dseek:     GOSUB Seek     ! Find the nearest point and draw a line
to it
2360            ON KEY #6 GOTO Da  ! Enable accept key
2370            ON KEY #7 GOTO Dr  ! Enable reject key
2380            GOTO 2380
2390 Da:        GOSUB Offkeys_a_r  ! Disable the accept and reject keys
2400            IF W=1 THEN P(W+1)=P(W)! Make sure the first pen code s
tays at -2
2410            FOR I=W TO N-1      ! Bump the rest of the coordinates d
own
2420            X(I)=X(I+1)
2430            Y(I)=Y(I+1)
2440            P(I)=P(I+1)
2450            NEXT I
2460            N=N-1
2470            X(N+1)=Y(N+1)=P(N+1)=0  ! Zero out the last coordinate
2480            GOTO Cancel
2490 Seek:      OFF KEY #14    ! Disable the seek key
2500            CALL Seek(X(*),Y(*),X,Y,N,W)     ! Find nearest point
2520            POINTER X,Y,2
2530            MOVE X,Y
```

```
2540            DRAW X(W),Y(W)
2550            RETURN                          ! Indicate the nearest point
2560 Dr:        GOSUB Offkeys_a_r
2570            GOSUB Undraw
2580            GOTO Deletepoint
2590 Insertpoint: IF N+1<=Nmax THEN 2620
2600            GOSUB Complain
2610            GOTO Point
2620            GOSUB Offkeys1          ! Disable command keys
2630            ON KEY #12 GOTO Cancel ! Enable the cancel and seek key
s
2640            ON KEY #14 GOTO Iseek1
2650            POINTER X,Y,1
2660            DIGITIZE X,Y
2670            GRAPHICS
2680            BEEP
2690            GOTO 2650
2700 Iseek1:    GOSUB Seekline          ! Find the nearest line and draw a
line to it
2710            ON KEY #6 GOTO Ia1   ! Enable the accept and reject keys
2720            ON KEY #7 GOTO Ir1
2730            GOTO 2730
2740 Ia1:       OFF KEY #12             ! Disable the cancel key
2750            GOSUB Offkeys_a_r    ! Disable the accept and reject key
s
2760            GOSUB Undrawline     ! Undraw the line to the nearest li
ne
2770            MOVE X(W1),Y(W1)     ! Undraw the line that was already
there.
2780            PEN -1
2790            DRAW X(W2),Y(W2)
2800            PEN 1
2810            DRAW Xmid,Ymid          ! Draw in two new lines
2820            DRAW X(W1),Y(W1)
2830            MOVE Xmid,Ymid
2840            FOR I=N TO W2 STEP -1
2850            X(I+1)=X(I)             ! Bump the points up one
2860            Y(I+1)=Y(I)
2870            P(I+1)=P(I)
2880            NEXT I
2890            N=N+1
2900            X(W2)=Xmid              ! Insert new coordinates
2910            Y(W2)=Ymid
2920            P(W2)=-1
2930            W=W2
2940            ON KEY #12 GOTO Cancel ! Re-enable the cancel key
2950            GOTO 1110    ! Branch into the middle of Movepoint rout
```

```
ine
2960 Ir1:       GOSUB Offkeys_a_r    ! Disable the accept and reject key
s
2970            GOSUB Undrawline     ! Undraw the line to the nearest li
ne
2980            GOTO 2640            ! Allow the user to select a differ
ent line
2990 Seekline:  OFF KEY #14          ! Disable the seek key
3000            CALL Lineseek(X(*),Y(*),X,Y,N,W1,W2,Xmid,Ymid)
3010            PEN 1
3020            POINTER X,Y,2        ! Use the blinking cross for a poin
ter
3030            MOVE X,Y             ! Draw a line to the middle of the
nearest
3040            DRAW Xmid,Ymid       !  line segment on the picture
3050            RETURN
3060 Undrawline: PEN -1              ! Undraw the line to the nearest li
ne
3070              MOVE X,Y
3080              DRAW Xmid,Ymid
3090              PEN 1
3100              MOVE X,Y
3110              RETURN
3120 Deleteline: GOSUB Offkeys1      ! Disable the command keys
3130            ON KEY #12 GOTO Cancel    ! Enable the cancel and seek k
eys
3140            ON KEY #14 GOTO Dlseek1
3150            POINTER X,Y
3160            DIGITIZE X,Y
3170            BEEP
3180            GRAPHICS
3190            GOTO 3150
3200 Dlseek1:   GOSUB Seekline       ! Find the nearest line segment
3210            ON KEY #6 GOTO Dla1  ! Enable the accept and reject keys
3220            ON KEY #7 GOTO Dlr1
3230            GOTO 3230
3240 Dla1:      GOSUB Offkeys_a_r    ! Disable the accept and reject key
s
3250            OFF KEY #12          ! Disable the cancel key
3260            GOSUB Undrawline
3270 Changepen:W=MAX(W1,W2)      ! Determine which pen code to change
3280            P(W)=-3-P(W)     ! If the pen code is -1 change it to -
2
3290                             ! If the pen code is -2 change it to -
1
3300            GOTO Cancel      ! Erase and re-draw the picture
3310 Dlr1:      GOSUB Offkeys_a_r
```

```
3320            GOSUB Undrawline
3330            GOTO 3140
3340 Savedata: EXIT GRAPHICS
3350            GOSUB Offallkeys ! Disable all keys
3360            INPUT "Which file do you want this figure kept in?",Fil
e$
3370            ASSIGN #1 TO File$,C
3380            IF C=1 THEN 3470
3390            BEEP
3400            DISP File$&" already exists.  Do you want to overwrite
it (Y/N)?";
3410            INPUT "",A$
3420            IF UPC$(A$)="N" THEN 3360
3430            IF UPC$(A$)="Y" THEN 3460
3440            BEEP
3450            GOTO 3400
3460            PURGE File$
3470            CREATE File$,N+1,30
3480            ASSIGN #1 TO File$
3490            PRINT #1,1;N
3500            READ #1,2
3510            FOR J=1 TO N
3520            PRINT #1,J+1;X(J),Y(J),P(J)
3530            NEXT J
3540            BEEP
3550            INPUT "Do you want to create another primitive (Y/N)?",
A$
3560            IF UPC$(A$)="Y" THEN 60
3570            IF UPC$(A$)="N" THEN 3600
3580            BEEP
3590            GOTO 3550
3600            ON Entrypoint GOTO 3620,3610
3610            LOAD "REPRO",Ep3
3620            END


**************************


3640 SUB Seek(X(*),Y(*),X,Y,N,W)
3650 ! This subprogram searches the arrays X(*) and Y(*) to find the c
oordinates
3660 !  closest to the point (X,Y).  N is the number of points in X(*)
 and Y(*).
3670 !  W is the subscript of the point that's closest.
3680 Mindis=9.99999999999E99
3690 W=0
```

```
3700 FOR I=1 TO N
3710 D=FNDis(I)
3720 IF D>=Mindis THEN Next    ! Compare the new distance against the m
inimum
3730                           !  known distance.
3740 W=I                       ! Update the subscript of the new minimu
m distance
3750 Mindis=D                  ! Update the new minimum distance
3760 Next: NEXT I
3770 SUBEXIT
3780 DEF FNDis(I)=SQR((X(I)-X)*(X(I)-X)+(Y(I)-Y)*(Y(I)-Y))
3790 SUBEND


************************


3810 SUB Lineseek(X(*),Y(*),X,Y,N,W1,W2,Xmid,Ymid)
3820 ! This subprogram searches all the line segments defined by the a
rrays
3830 !  X(*) and Y(*) for the one closest to the point (X,Y).  N is th
e number
3840 !  of points in the arrays, and W1 and W2 define the endpoints of
 the
3850 !  nearest segment. Xmid and Ymid are the midpoints of the neares
t segment.
3860 Mindis=9.99999999999E99
3870 FOR I=1 TO N-1
3880 A=Y(I+1)-Y(I)         ! Find equation of line between points I and
I+1 in the
3890 B=X(I)-X(I+1)         !  form:  Ax + By + C = 0
3900 C=-X(I)*A-Y(I)*B
3910 ON ERROR GOTO Bomb
3920 Distance=ABS(A*X+B*Y+C)/SQR(A*A+B*B)  ! Distance formula from poi
nt to line
3930 Xhat=(B*B*X-A*B*Y-A*C)/(A*A+B*B)    ! (Xhat,Yhat) is the point on
the line
3940 Yhat=(-A*B*X+A*A*Y-B*C)/(A*A+B*B)   !  which is closest
3950 OFF ERROR
3960 !  Now check to see if the intersection point (Xhat,Yhat) lies be
tween the
3970 !  endpoints of the line segment.
3980 IF X(I+1)=X(I) THEN Checky
3990 Checkx:  IF (X(I)<=Xhat) AND (Xhat<=X(I+1)) THEN Okay
4000          IF (X(I+1)<=Xhat) AND (Xhat<=X(I)) THEN Okay
4010          GOTO Endpoints
4020 Checky:  IF (Y(I)<=Yhat) AND (Yhat<=Y(I+1)) THEN Okay
```

```
4030          IF (Y(I+1)<=Yhat) AND (Yhat<=Y(I)) THEN Okay
4040 Endpoints:   ! Compute the distances to the endpoints of the segm
ent since
4050              ! the perpendicular distance is to a point not on th
e segment.
4060           D1=FND(I)
4070           D2=FND(I+1)
4080           Distance=MIN(D1,D2)
4090 Okay:  IF Distance>=Mindis THEN Nexti    ! Compare new distance wi
th minimum
4100                                          !  known distance.
4110        Mindis=Distance          ! Update new minimum distance.
4120        W1=I                     ! Update the subscripts of the end
points of
4130        W2=I+1                   !  the nearest known segment.
4140        Xmid=(X(I)+X(I+1))/2     ! Update the midpoint coordinates
of the
4150        Ymid=(Y(I)+Y(I+1))/2     !  nearest known line segment.
4160 Nexti:  NEXT I
4170 SUBEXIT
4180 DEF FND(J)=SQR((X(J)-X)^2+(Y(J)-Y)^2)
4190 Bomb: IF ERRN=31 THEN Nexti
4200 BEEP
4210 EXIT GRAPHICS
4220 PRINT ERRM$
4230 PAUSE
4240 SUBEND
```

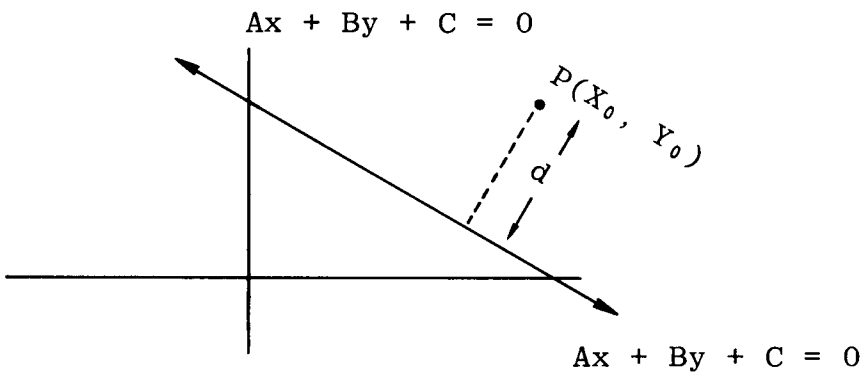
Methods and Formulae:

Distance between two points $(x_0, y_0)$ and $(x_1, y_1)$

$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

Distance from a point to a line:

given the point has coordinates $(x_0, y_0)$
and the line has the equation




$$d = \frac{| Ax_0 + By_0 + C |}{\sqrt{A^2 + B^2}}$$

REFERENCES:

1. Protter/Morrey, College Calculus with Analytical Geometry, Second Edition (Addison Wesley, 1970) p. 281.

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary transport (i.e., the transport above the special function keys)
2. Load the file:
   a. Type: LOAD "CONSTR: T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Do you want to use an old file to begin with (Y/N)?" appears in the display area:
   a. If you want to use an old picture file (i.e., a picture that has been saved previously):
      1) Type: Y
      2) Press: CONT
      3) Go to step 5

      or
   a. If you do not want to use an old picture file:
      1) Type: N
      2) Press: CONT
      3) Go to step 6
5. When "Which file do you wish to use?" appears in the display area:
   a. Type: The name of the file you wish to use
   b. Press: CONT
   c. Go to step 6
   d. If the message "File is not available" appears in the display area, then the file you entered either does not exist on the present default mass storage device, or else the file exists, but is not a data file. In this case either enter a different file name, or specify a different default mass storage device (use the MASS STORAGE IS command - see the operating and programming manual for details). Go back to step 4 in this case.

e. If the message "FILE TOO LARGE" appears in the display area, then the number of points in the file exceeds Nmax, (see the section under Variables (Main Program)) the maximum number of points allowed in memory. If this happens, go back to step 4. Also, refer to the section under Special Considerations and Programming Hints.

6. The 9845 will go into graphics mode and a set of crosshairs will appear on the screen. The crosshairs may be moved by using the keys marked DISPLAY on the 9845 keyboard (left arrow, up arrow, right arrow, down arrow). At this point the following special function keys may be pressed to issue commands.

Key 8 - draw a line to the present position of the crosshairs

Key 10 - move to the present position of the crosshairs (without drawing a line)

Key 3 - move a point

Key 4 - delete a point

Key 5 - insert a point

Key 13 - invert a line (if a line is on (visible), turn it off (make it invisible), or vice-versa)

Key 15 - Save the present picture in a data file

a. Press: One of the special function keys listed above (8, 10, 3, 4, 5, 13, 15)

b. Go to one of the following steps:

1) If you pressed Key 8, go to step 7
2) If you pressed Key 10, go to step 8
3) If you pressed Key 3, go to step 9
4) If you pressed Key 4, go to step 11
5) If you pressed Key 5, go to step 12
6) If you pressed Key 13, go to step 13
7) If you pressed Key 15, go to step 14

Note: If, as a result of pressing keys 8, 10, or 5, the message "ARRAY OVERFLOW--NO MORE POINTS CAN BE STORED" appears in the display, the X(*), Y(*), and P(*) arrays are full and will hold no more points. After this

message goes away, go back to step 6. Refer to the section under Special Considerations and Programming Hints.

7. (Draw) A line will be drawn from the previous position of the pen to the present position of the crosshairs. When the line has been drawn, go back to step 6a. Note: If this is the first point on the picture, Key 8 will give the same effect as Key 10.
8. (Move) The pen will be moved from its previous position to the current position of the crosshairs. Return to step 6a.
9. (Move a point by "rubber-banding") Once Key 3 has been pressed, position the crosshairs near the point you wish to move.

   Note: Key 12 may be pressed to cancel the move point operation at any time. If key 12 is pressed, return to step 6a.

   a. Once the crosshairs are positioned:
      1) Press: Key 14 (seek the nearest point)
   b. A blinking cross will appear, and a line will be drawn from the cross to the point which the program assumes you want to move.
      1) If the indicated point is the point you wish to move:
         a) Press: Key 6 (accept the indicated point)
         b) Go to step 10.

         or

      1) If the indicated point is <u>not</u> the point you wanted to move:
         a) Press: Key 7 (reject the indicated point)
         b) Reposition the crosshairs so that they are closer to the desired point.
         c) Go to step 9a.
10. The blinking cross will appear at the point indicated in step 9. Now the point may be moved by pressing the following keys:

    Key 0 - left
    Key 1 - up
    Key 2 - right
    Key 9 - down

    Finer increments may be obtained by shifting the given keys.

    Note: For quick repetition of a keystroke, hold down on the

REPEAT key at the same time as you are holding down the Special Function Key.

a. When the point has been positioned where you want it:
   1) Press: Key 11 (set the point)
   2) Go back to step 6a.

11. (Delete a point) Once key 4 has been pressed, position the crosshairs near point you wish to remove.

Note: Key 12 may be pressed to cancel the delete point operation at any time. If Key 12 is pressed, return to step 6a.

a. Once the crosshairs are positioned:
   1) Press: Key 14 (seek the nearest point).

b. A blinking cross will appear, and a line will be drawn from the cross to the point which the program assumes you want to delete.
   1) If the indicated point is the point you wish to delete:
      a) Press: Key 6 (accept the indicated point)
      b) Go to step 6a.

      or

   1) If the indicated point is not the point you meant to delete:
      a) Press: Key 7 (reject the indicated point)
      b) Reposition the crosshairs so that they are closer to the desired point.
      c) Go to step 11a.

12. (Insert a Point) This operation allows the user to select a line segment where a point may be inserted. The inserted point will be at the middle of the line segment. To position the new point, the "rubber-banding" mode is used.

Note: The insert point operation may be cancelled by pressing Key 12 <u>prior</u> to pressing the accept Key (Key 6). Once the line segment where the insertion is to occur has been accepted, the operation progresses from insertion to "rubber-banding". At this stage, Key 12 will cancel the rubber-band mode and leave the inserted

point at the midpoint of the accepted line segment.

a. Position the crosshairs near the line segment where you want to insert the point.

b. Press: Key 14 (seek the nearest line segment)

c. A blinking cross will appear and a line will be drawn from the cross to the midpoint of the nearest line segment.

1) If the indicated line segment is the line segment where you want to insert the point:

a) Press: Key 6 (accept)

b) Go to step 10.

or

1) If the indicated line segment is not the line segment where you want to insert the point:

a) Press: Key 7 (reject)

b) Reposition the crosshairs so that they are closer to the desired line segment.

c) Go to step 12a.

13. (Invert a line) This operation allows the user to invert the status of a line segment. That is, a line that is visible may be made invisible, and a line that is invisible may be made visible.

Note: This operation may be cancelled by pressing Key 12 at any time.

a. Position the crosshairs near the line segment you want to invert.

b. Press: Key 14 (seek the nearest line segment)

c. A blinking cross will appear and a line will be drawn from the cross to the midpoint of the nearest line segment.

1) If the indicated line segment is the line segment which you want to invert:

a) Press: Key 6 (accept)

b) Go to step 6a.

or

1) If the indicated line segment is not the line segment

you want to invert:

a) Press: Key 7 (reject)

b) Reposition the crosshairs so that they are nearer the line segment you want to invert

c) Go to step 13a.

14. (Save data) When "Which file do you want this figure kept in?" appears in the display area:

a. Type: A valid file name (a file name may be up to six characters long)

b. Press: CONT

15. If "*file* already exists. Do you want to overwrite it (Y/N)?" appears in the display area of the CRT:

a. If you want to overwrite the specified file:

1) Type: Y

2) Press: CONT

3) Go to step 16.

or

a. If you do not want to overwrite the specified file:

1) Type: N

2) Press: CONT

3) Go to step 14.

16. When "Do you want to create another primitive (Y/N)?" appears in the display area:

a. If you want to draw another picture:

1) Type: Y

2) Press: CONT

3) Go to step 6.

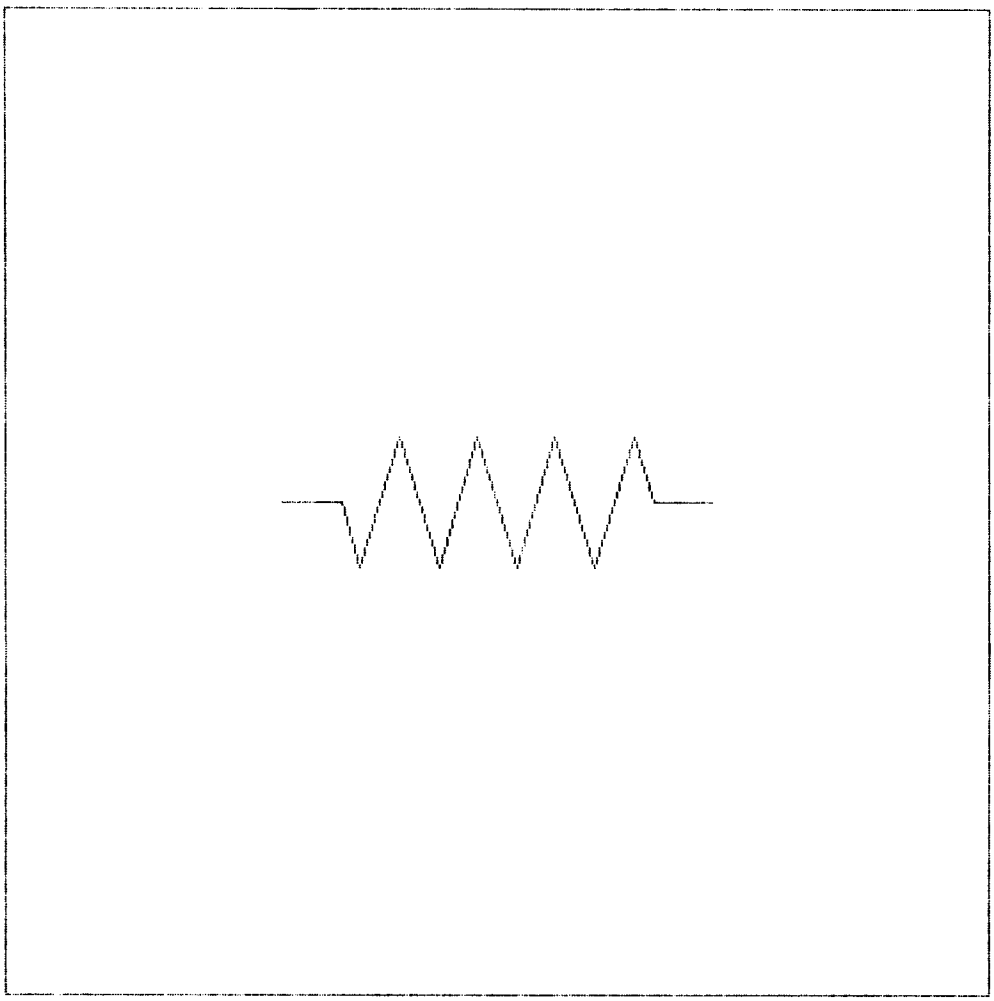
or

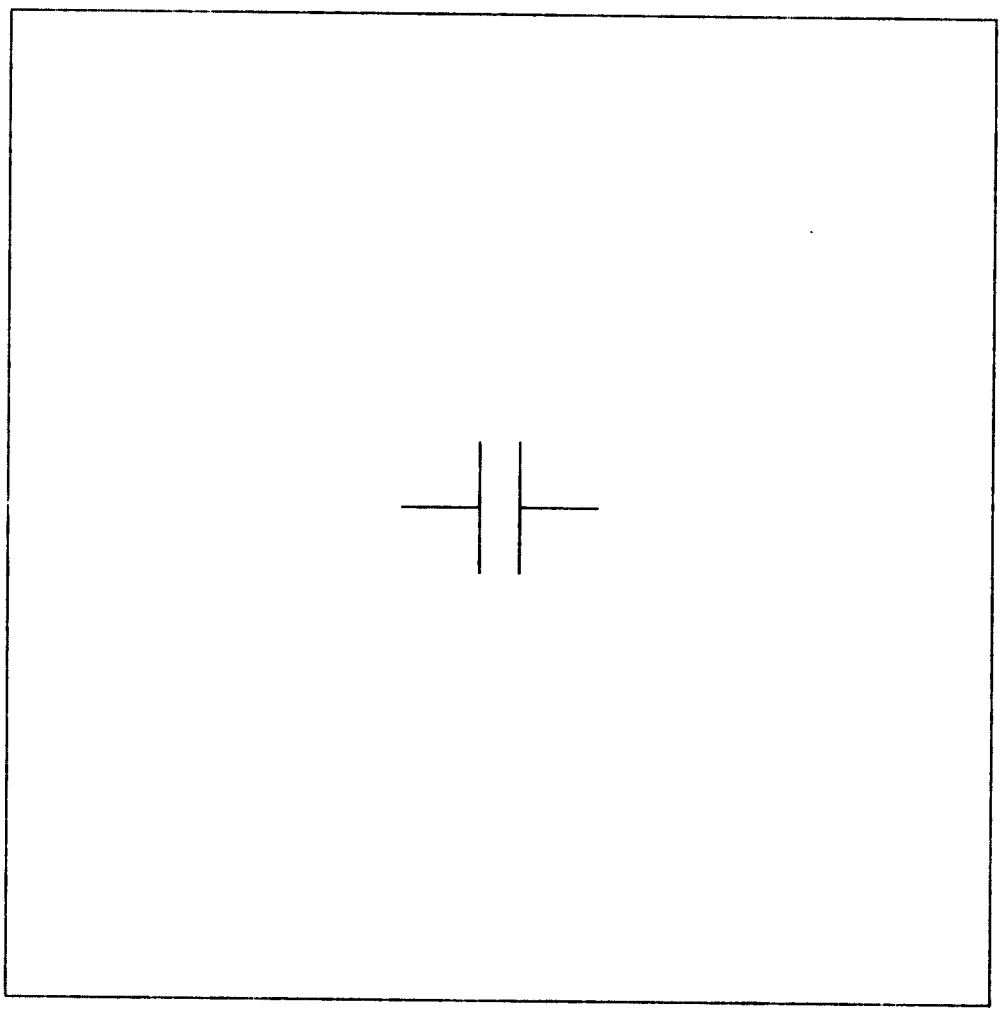
a. If you do not want to draw another picture:

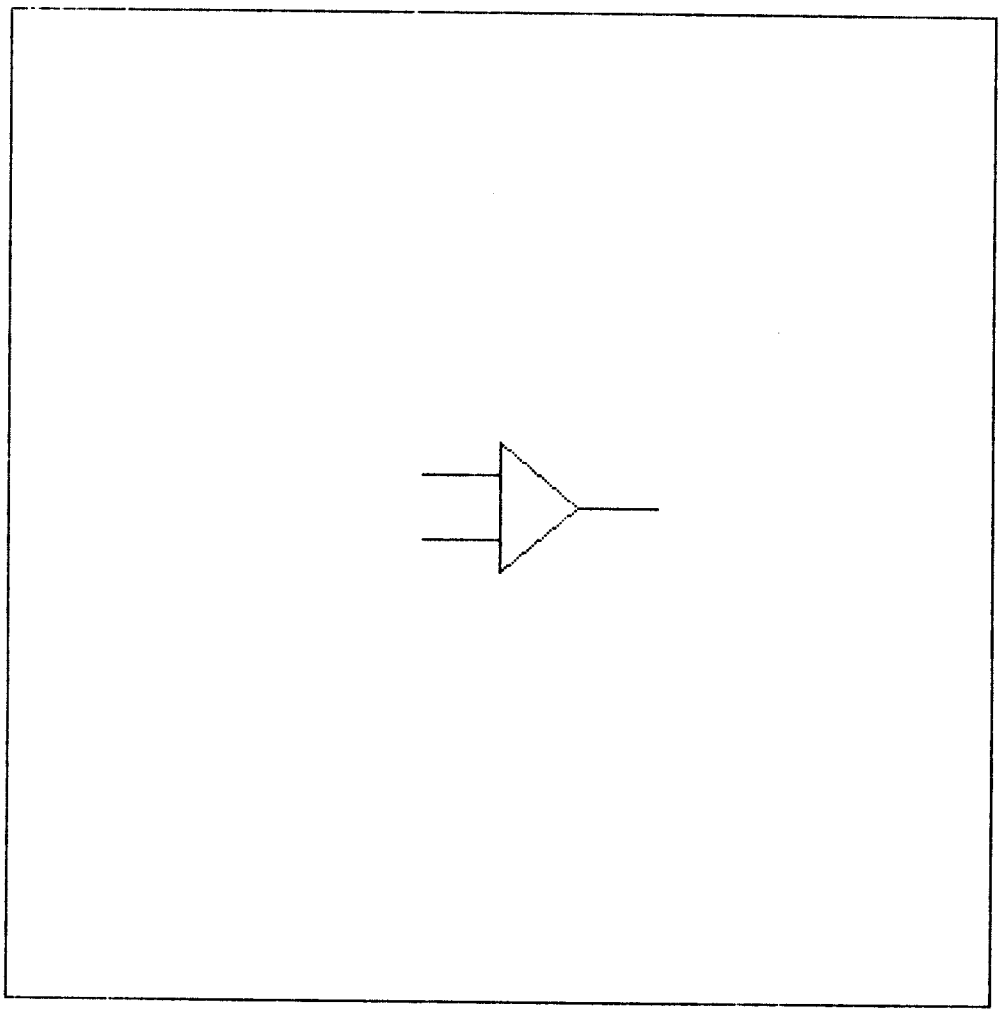
1) Type: N

2) Press: CONT

3) The program stops.







HEWLETT hp PACKARD

# ENTITY CREATION PROGRAM

This program is similar in nature to the picture construction program, only it operates at a higher level. Where the picture construction program deals in lines and points, this program deals in figures, or "primitives". The picture construction program is used to construct "primitives" which become the building blocks that this program uses.

The user may specify up to 10 "primitives" for use with the program. The "primitives" are arranged in a menu. The user may select "primitives", position them anywhere on the screen, scale the "primitives", and rotate them in any way he chooses.

In addition, the entities constructed with this program may in turn be converted into "primitives" to be used in creating still more complex "primitives".

Program Utilization:

File Name: REPRO

Subprograms Required:

Drawfig
Menu
Movefigure
Primitive
Repro
Seek

Variables (Main Program):

A$ - Used for user's answer to Y/N questions

C - Used with ASSIGN statements to see whether or not a file is available

Delete - The subscript of the menu item to be deleted

| | | |
|---|---|---|
| Entry point | - | Tells where program execution began |
| File$ | - | Holds file name where data will be stored |
| File$(*) | - | Array holding the menu items |
| I | - | Loop counter |
| J | - | Loop counter |
| N | - | The number of points in a "primitive" |
| Nofigures | - | The number of figures, or "primitives", in the menu |
| No pictures | - | The number of figures that make up the picture |
| Pictures(*) | - | This array holds the transformations of the figures that make up the entity |
| Temp | - | A temporary variable passed to a subprogram |
| W | - | Tells which Figure, or "primitive", in the entity which is being affected |
| Which | - | Tells which figure, or "primitive", has been selected from the mean |
| X | - | X Coordinate of digitized point |
| Y | - | Y Coordinate of digitized point |

Variables (Subprogram Repro):

| | | |
|---|---|---|
| I | - | Loop counter |
| N | - | The number of points in the figure being reproduced |
| P | - | The pen code (1 or 2 for draw, -1 for erase |
| Pictures(*) | - | The array holding the transformations of the figures on the screen |
| T | - | Tells which figure, or "primitive", in the main entity is being drawn |
| X(*) | - | The array holding the points being drawn |

Variables (Subprogram Primitive):

| | | |
|---|---|---|
| I | - | Loop counter |
| N | - | The number of points in the "primitive" file |
| W | - | Tells which primitive is being drawn |
| X(*) | - | The array containing the points to be drawn |

Variables (Subprogram Seek):

| | | |
|---|---|---|
| D | - | Distance to each figure |
| I | - | Loop counter |
| Mindis | - | The distance to the nearest figure |
| N | - | The number of "primitives" in the entity |
| Pictures(*) | - | The transformation array |
| W | - | Tells which "primitive" is closest |
| X | - | X coordinate of the point from which all distances are measured |
| Y | - | Y coordinate of the point from which all distances are measured |

Variables (Subprogram Movefigure):

| | | |
|---|---|---|
| H1 | - | Old X coordinate of the location of the "primitive" (saved in case the move operation is cancelled) |
| H2 | - | Old Y coordinate of the location of the "primitive" (saved in case the move operation is cancelled) |
| Increment | - | Tells how far the figure will be moved with each keystroke |
| N | - | The number of points in the figure, or "primitive", being used |
| Pictures(*) | - | The transformation array |
| Sideinc | - | Tells how far to move the figure horizontally |
| Upinc | - | Tells how far to move the figure vertically |

W - Tells which "primitive" in the main picture is being moved

X(*) - The array containing the points of the figure being moved

Variables (Subprogram Drawfig):

I - Loop counter

N - The number of points in the figure being drawn

P - Pen code (1 or 2 for draw, -1 for erase)

Pictures(*) - The transformation array

T - Tells which figure is being drawn

X(*) - The array containing the points of the figure being drawn

Variables (Subprogram Menu):

I - Loop counter

J - Tells which menu item is being drawn

N - Tells the number of points in the menu item

X(*) - The array holding the point to be drawn

Special Considerations and Programming Hints:

1. System Configuration:
   Standard Memory Option
   CRT Graphics Hardware
   Graphics ROM
   (Optional) Internal Thermal Printer (for CRT graphics dumps)

# Annotated Listing of Entity Reproduction Program

```
10   Entrypoint=1
20   Nopictures=0
30   OPTION BASE 1
40   DIM File$(10),Pictures(100,7)  !  Pictures(I,1) is menu number (w
hich one)
50                                  !  Pictures(I,2) is rotation angle
 (degrees)
60                                  !  Pictures(I,3) is menu X
70                                  !  Pictures(I,4) is menu Y
80                                  !  Pictures(I,5) is real X (to men
u X)
90                                  !  Pictures(I,6) is real Y (to men
u Y)
100                                 !  Pictures(I,7) is scale factor
110 Setplotter:   PLOTTER IS "GRAPHICS"
120  LIMIT 35,165,10,140
130  SCALE -5,5,-5,5
140  ON Entrypoint GOTO 200,Draw_with_menu
150                                 !  There are two entrypoints to th
e program:
160                                 !   It may either start with RUN,
or it may
170                                 !   be re-loaded by the conversion
 program
180                                 !   (starts at Ep3 and branches ba
ck to
190                                 !   Setplotter).
200  PRINTER IS 16
210  PRINT PAGE;"   This program allows you to build a picture by digi
tization and"
220  PRINT "menu selection.  You may either start fresh, or you may bu
ild on an already"
230  PRINT "existing file."
240  PRINT "   This program will be easier to use if you take a blank
overlay"
250  PRINT "(part no. 7120-6164), and fill in the appropriate squares
as follows:",LIN(1)
260  PRINT "KEY 0:  <-- (left arrow)";TAB(30);"KEY 8:  PICK FIGURE"
270  PRINT "KEY 1:   ^  (up arrow)";TAB(30);"KEY 10: DRAW LINE"
280  PRINT "KEY 2:  --> (right arrow)";TAB(30);"KEY 3:  MOVE FIGURE";T
AB(55);"KEY 11: SET FIGURE"
290  PRINT "KEY 9:      (down arrow)";TAB(30);"KEY 4:  DELETE FIGURE"
300  PRINT TAB(30);"KEY 5:  DRAW PICTURE WITH MENU"
310  PRINT TAB(30);"KEY 13: DRAW PICTURE WITHOUT MENU"
320  PRINT TAB(30);"KEY 24: MODIFY MENU"
330  PRINT TAB(30);"KEY 26: CONVERT PICTURE TO PRIMITIVE"
340  PRINT TAB(30);"KEY 27: CONSTRUCT PRIMITIVE"
```

```
350  PRINT LIN(1),TAB(45);"KEY 14: SEEK",LIN(1);TAB(20);"KEY 12: CANCE
L";TAB(45);"KEY 6:  ACCEPT";LIN(1),TAB(45);"KEY 7:  REJECT"
360  PRINT "KEY 15: SAVE PICTURE"
370  INPUT "Do you want to use an old picture configuration (Y/N)?",A$
380  IF UPC$(A$)="N" THEN Startfresh
390  IF UPC$(A$)="Y" THEN Use_old_file
400  BEEP
410  GOTO 370
420 Use_old_file:  INPUT "Please enter the filename of the old picture
",File$
430       ASSIGN #1 TO File$,C
440       IF NOT C THEN File_available
450       BEEP
460       DISP File$&" is not available"
470       WAIT 1000
480       GOTO 370
490 File_available:  READ #1;Nopictures,Nofigures
500                               ! Nopictures is the number of discrete
 entities
510                               !  which appear on the screen
520                               ! Nofigures is the number of primitive
s
530                               !  resident in the user's library.
540       READ #1,2
550       READ #1;Pictures(*),File$(*)
560                               ! Pictures(*) contains the transformat
ions
570                               !  of each entity.
580                               ! File$(*) contains the file names of
the
590                               !  raw data points of the primitives.
600       FOR I=1 TO Nofigures
610       ASSIGN #I TO File$(I)  ! Assign and buffer the primitive file
s.
630       NEXT I
640       GOTO Draw_with_menu
650 Startfresh:  INPUT "How many figures do you want in your library?"
,Nofigures
660       Nofigures=INT(Nofigures)
670       IF (Nofigures>=0) AND (Nofigures<=10) THEN 730
680                    ! There can be only ten primitives because 1) th
ere are
690                    !  only ten file pointers, and 2) there is only
enough
700                    !  room across the screen for ten menu members.
710       BEEP
720       GOTO 650
```

```
730       FOR I=1 TO Nofigures
740       DISP "Please enter the filename of figure #"&VAL$(I);
750       INPUT "",File$(I)
760       ASSIGN #I TO File$(I),C    ! Assign the file
770       IF NOT C THEN 830
780       BEEP
790       DISP "File "&File$(I)&" is not available"
800       WAIT 1000
810       GOTO 740
830       NEXT I
840       GOTO Draw_with_menu
850 Modlib: EXIT GRAPHICS            ! Modify the primitives library
860         GOSUB Offkeys
870         INPUT "Do you want to add figures to the library, or delet
e them (A/D)?",A$
880         IF UPC$(A$)="A" THEN Insertmenu
890         IF UPC$(A$)="D" THEN Deletefrommenu
900         BEEP
910         GOTO 870
920 Deletefrommenu: INPUT "Please enter the number of the figure you w
ish to delete from the library",Delete
930         Delete=INT(Delete)
940         IF (Delete>0) AND (Delete<=10) THEN Delete
950         BEEP
960         DISP "There is no figure number "&VAL$(Delete)
970         WAIT 1000
980         GOTO 1060
990 Delete: FOR I=Delete TO Nofigures-1
1000        File$(I)=File$(I+1)
1010        NEXT I
1020        Nofigures=Nofigures-1
1030        BEEP
1040        DISP "Figure deleted."
1050        WAIT 1000
1060        INPUT "Do you want to delete more figures (Y/N)?",A$
1070        IF UPC$(A$)="N" THEN Re_assign
1080        IF UPC$(A$)="Y" THEN 1080
1090        BEEP
1100        GOTO 920
1101 Re_assign: FOR I=1 TO Nofigures
1102        ASSIGN #I TO File$(I)
1103        NEXT I
1104        FOR I=Nofigures+1 TO 10
1105        ASSIGN #I TO *
1106        NEXT I
1109        GOTO Draw_with_menu
1110 Insertmenu:IF Nofigures<10 THEN 1160
```

```
1120         BEEP
1130         DISP "No more room to insert figures."
1140         WAIT 1000
1150         GOTO Draw_with_menu
1160         DISP "Please enter the filename of figure #"&VAL$(Nofigure
s+1);
1170         INPUT "",File$(Nofigures+1)
1180         ASSIGN #Nofigures+1 TO File$(Nofigures+1),C
1190         IF C=0 THEN 1240
1200         BEEP
1210         DISP File$(Nofigures+1)&" is not available"
1220         WAIT 1000
1230         GOTO 1250
1240         Nofigures=Nofigures+1
1250         INPUT "Do you want to add more figures to the library (Y/N
)",A$
1260         IF UPC$(A$)="N" THEN Draw_with_menu
1270         IF UPC$(A$)="Y" THEN Insertmenu
1280         BEEP
1290         GOTO 1250
1300 Draw_with_menu: GRAPHICS
1310                 GCLEAR
1320                 GOSUB Drawmenu
1330 Draw_no_menu:   GRAPHICS
1340                 GOSUB Drawpictures
1350 ON KEY #8 GOTO Pick        ! Enable keys
1360 ON KEY #24 GOTO Modlib
1370 ON KEY #5 GOTO Draw_menu
1380 ON KEY #13 GOTO Draw_clear
1390 ON KEY #15 GOTO Save
1400 ON KEY #10 GOTO Line
1410 ON KEY #3 GOTO Movefigure
1420 ON KEY #4 GOTO Deletefigure
1430 ON KEY #26 GOTO Convert
1440 ON KEY #27 GOTO Construct
1450 GRAPHICS                   ! Sit idle waiting for a command
1451 GOTO 1450
1460 Pick:  GOSUB Offkeys       ! Disable keys
1470         ON KEY #12 GOTO Cancel
1480         SETGU              ! Set GDU's
1500         POINTER 10*Nofigures+5,95,2 ! Position the pointer after t
he last
1510                                     !  menu item.
1520         DIGITIZE X,Y
1530         PEN 1
1540         IF (Y<90) OR (Y>100) THEN 1350   ! Test for out of bounds
1550         IF (X<0) OR (X>Nofigures*10) THEN Draw_with_menu
```

```
1560          Which=INT(X/10)+NOT (NOT FRACT(X/10))+NOT X
1561                                                  ! Compute which figure wa
s selected
1570          Pictures(Nopictures+1,1)=Which    ! Store the file number i
n the
1580                                                  !  transformation array f
or the
1590                                                  !  entity.
1600          READ #Which,1;N                         ! Read the number of poin
ts which
1610                                                  !  which make up the new
primitive.
1620          CALL Primitive(#Which,Which,N)    ! Blow the figure up
1630          POINTER .5,.5
1640          DIGITIZE Pictures(Nopictures+1,3),Pictures(Nopictures+1,4)
1650                                  ! Digitize a reference point on the
primitive.
1660          GCLEAR
1670          GOSUB Drawpictures      ! Put the picture back on the scree
n
1680          POINTER 0,0
1690          DIGITIZE Pictures(Nopictures+1,5),Pictures(Nopictures+1,6)
1700                                  ! Digitize the point in the main pic
ture where
1710                                  !  the new entity will be positioned
1720          GCLEAR
1730          Nopictures=Nopictures+1
1740          EXIT GRAPHICS
1750 Angle:   INPUT "What angle of rotation?",Pictures(Nopictures,2)
1760 Scale:   INPUT "Scale (1-10)?",Pictures(Nopictures,7)
1770                                     ! Enter the transformations for t
he new
1780                                     !  entity.
1790          IF (Pictures(Nopictures,7)>=1) AND (Pictures(Nopictures,7)
<=10) THEN 1820
1800          BEEP
1810          GOTO Scale
1820          GRAPHICS
1830          GOTO Draw_with_menu       ! Re-draw the picture and wait fo
r command.
1840 Line:    GOSUB Offkeys             ! Disable keys
1850          Nopictures=Nopictures+1
1860          Pictures(Nopictures,1)=11
1870          PEN 1
1880          POINTER 0,0
1890          DIGITIZE Pictures(Nopictures,3),Pictures(Nopictures,4)
1900          MOVE Pictures(Nopictures,3),Pictures(Nopictures,4)
```

```
1910         GRAPHICS
1920         POINTER Pictures(Nopictures,3),Pictures(Nopictures,4)
1930         DIGITIZE Pictures(Nopictures,5),Pictures(Nopictures,6)
1940         GRAPHICS
1950         DRAW Pictures(Nopictures,5),Pictures(Nopictures,6)
1960                 ! Digitize the endpoints of the line, and draw it
in.
1970         GOTO 1350 ! Enable the keys again and wait for the next co
mmand.
1980 Movefigure:  IF Nopictures THEN 2020     ! Check to see if there's
 a picture
1990                                          !  to move
2000               BEEP
2010               GOTO 1450
2020               GOSUB Offkeys     ! Disable previous keys
2021               X=Y=0
2030               ON KEY #12 GOTO Cancel     ! Enable Cancel and Seek
keys.
2040               ON KEY #14 GOTO Mseek
2050               POINTER X,Y
2060               DIGITIZE X,Y
2070               GRAPHICS
2080               BEEP
2090               GOTO 2050
2100 Mseek:  GOSUB Seek        ! Disable the Seek key and find the near
est figure
2110          ON KEY #6 GOTO Ma   ! Accept
2120          ON KEY #7 GOTO Mr   ! Reject
2130          GRAPHICS
2135          GOTO 2130           ! Wait for acceptance, rejection or ca
ncellation
2140 Ma:      GOSUB Offkeys67     ! Disable Accept and Reject keys
2141          OFF KEY #12
2150          GOSUB Undraw        ! Undraw the line to the nearest figur
e.
2151          IF Pictures(W,1)<>11 THEN 2160
2152          CALL Movefigure(#1,1,W,Pictures(*))
2153          GOTO Cancel
2160          READ #Pictures(W,1),1;N
2170          CALL Movefigure(#Pictures(W,1),N,W,Pictures(*))
2180 Cancel:     GCLEAR
2190             GOTO Draw_with_menu
2200 Mr: GOSUB Offkeys67         ! Disable Accept and Reject keys.
2210     GOSUB Undraw            ! Undraw the line to the nearest figur
e.
2220     GOTO 2040               ! Allow the user to select a different
 figure
```

```
2230 Offkeys67: OFF KEY #6      ! This subroutine disables the Accept
and Reject
2240            OFF KEY #7      !  keys.
2250            RETURN
2260 Deletefigure:  IF Nopictures THEN 2300  ! Check to see if there's
 a figure
2270                                         !  to delete.
2280                BEEP
2290                GOTO 1450
2300                GOSUB Offkeys   ! Disable previous keys
2301                X=Y=0
2310                ON KEY #12 GOTO Canceldelete ! Enable Cancel and S
eek keys.
2320                ON KEY #14 GOTO Dseek
2330                POINTER X,Y,1
2340                DIGITIZE X,Y
2350                GRAPHICS
2360                BEEP
2370                GOTO 2330
2380 Dseek: GOSUB Seek          ! Disable the Seek key and find the near
est figure
2390        ON KEY #6 GOTO Da ! Accept
2400        ON KEY #7 GOTO Dr ! Reject
2410        GOTO 2410         ! Wait for acceptance, rejection, or can
cellation
2420 Da:    GOSUB Offkeys67   ! Disable accept and reject keys.
2430        OFF KEY #12       ! Disable Cancel key
2440        FOR I=W TO Nopictures-1
2450        FOR J=1 TO 7
2460        Pictures(I,J)=Pictures(I+1,J) ! Bump the rest of the trans
formations
2470        NEXT J                        !  down one.
2480        NEXT I
2490        Nopictures=Nopictures-1       ! There is now one less enti
ty.
2500        GOTO Draw_with_menu           ! Re-draw the screen
2510 Dr:    GOSUB Offkeys67     ! Disable accept and reject keys
2520        GOSUB Undraw        ! Undraw the line to the nearest figur
e
2530        GOTO 2320           ! Allow the user to select a different
 figure
2540 Canceldelete: GOSUB Offkeys67  ! Disable accept and reject keys
2550               OFF KEY #12      ! Disable cancel key
2560               GOTO Draw_with_menu
2570 Draw_clear:GCLEAR
2580            GOTO Draw_no_menu
2590 Draw_menu: GCLEAR
```

```
2600              GOTO Draw_with_menu
2610 Drawmenu:    FOR I=1 TO Nofigures      ! Draw the menu
2620              READ #I,1;N               ! Read in the number of points
 in the
2630              CALL Menu(#I,I,N)         !  ith menu item and call the
subprogram
2640              NEXT I                    !  to draw that item.
2650              RETURN
2660 Drawpictures: LOCATE 0,100,0,100       ! Reset screen boundaries for
main
2670              SETGU                     !  picture.
2671              FRAME
2680              SCALE -5,5,-5,5
2690              FOR I=1 TO Nopictures
2700              IF Pictures(I,1)=11 THEN Drawline
2710                                        ! Check to see if the next ent
ity is a
2720                                        !  simple line, or a primitive
 that has
2730                                        !  been put through some trans
formations
2740              READ #Pictures(I,1),1;N
2750              Temp=I
2760              CALL Repro(#Pictures(I,1),N,Temp,Pictures(*),1)
2770                       ! The subprogram Repro will put the primitive
 indicated
2780                       !  by Pictures(I,1) through the given transfo
rmations.
2790              GOTO 2820
2800 Drawline:    MOVE Pictures(I,3),Pictures(I,4)  ! Draw the indicated
 line.
2810              DRAW Pictures(I,5),Pictures(I,6)
2820              NEXT I
2830              RETURN
2840 Offkeys:  OFF KEY #8      ! Disable the main command keys.
2850           OFF KEY #24
2860           OFF KEY #10
2870           OFF KEY #26
2880           OFF KEY #15
2890           OFF KEY #27
2900           OFF KEY #3
2910           OFF KEY #4
2920           OFF KEY #5
2930           OFF KEY #13
2940           RETURN
2950 Convert:  ASSIGN #1 TO "!@#$%^",C     ! "!@#$%^" is a special file
used by
```

```
2960          IF C=1 THEN 2980              ! this program and the conver
sion
2970          PURGE "!@##%^"                ! program for communication.
 This
2980          CREATE "!@##%^",105,70        ! avoids having to ask the us
er
2990          ASSIGN #1 TO "!@##%^"         ! questions.
3000          PRINT #1,1;Nopictures,Nofigures
3010          READ #1,2
3020          PRINT #1;Pictures(*),File$(*)
3030          LOAD "CONVRT",Ep2               ! Load the conversion progr
am and
3040                                        !  run it.

 ************************


3060 Construct: ASSIGN #1 TO "!@##%^",C   ! This segment of the program
 allows
3070          IF C=1 THEN 3090              !  user to load in the primit
ive
3080          PURGE "!@##%^"                !  construction program and c
onstruct
3090          CREATE "!@##%^",105,70        !  a new primitive and then b
ranch
3100          ASSIGN #1 TO "!@##%^"         !  back into this program and
 pick up
3110          PRINT #1,1;Nopictures,Nofigures ! where he left off.
3120          READ #1,2
3130          PRINT #1;Pictures(*),File$(*)
3140          LOAD "CONSTR",Ep2


 ************************


3160 Ep3:  ASSIGN #1 TO "!@##%^"            ! This segment of the progra
m is for
3170       READ #1,1;Nopictures,Nofigures !  linking back into this pr
ogram from
3180       READ #1,2                       !  either the conversion pro
gram, or
3190       READ #1;Pictures(*),File$(*)    !  the primitive constructio
n program.
3200       Entrypoint=2                    ! Set a flag so that the pro
gram can
```

```
3210                                    !  tell at a later point whi
ch entry
3220                                    !  point was used.
3230        FOR I=1 TO Nofigures
3240        ASSIGN #I TO File$(I)       ! Assign and buffer the file
s in the
3250        BUFFER #I                   !  primitive library.
3260        NEXT I
3270        GOTO Setplotter             ! Set the screen boundaries
and scales


 ************************


3290 Seek: OFF KEY #14                  ! Disable cancel and seek ke
ys.
3300        OFF KEY #12
3310        CALL Seek(Pictures(*),X,Y,Nopictures,W)
3320                                    ! Find the nearest figure an
d return
3330                                    !  its reference point in (X
,Y).
3350        POINTER X,Y,2               ! Use the blinking cross for
 POINTER
3360        MOVE X,Y
3370        DRAW Pictures(W,5),Pictures(W,6) ! Draw a line to the neare
st figure.
3371        ON KEY #12 GOTO Cancel
3380        RETURN
3390 Undraw: MOVE X,Y
3400        PEN -1
3410        DRAW Pictures(W,5),Pictures(W,6) ! Undraw the line to the n
earest
3420                                      !  figure.
3430        PEN 1                         ! Reset the pen
3440        RETURN


 ************************


3460 Save: EXIT GRAPHICS      ! This segment saves the picture configur
ation
3470                          !  in a file of the user's choice.
3480        GOSUB Offkeys     ! Disable the keys.
3490        INPUT "Which file do you want to use to store the picture c
```

```
onfiguration?",File$
3500        ASSIGN #1 TO File$,C
3510        IF C=1 THEN 3600
3520        BEEP
3530        DISP File$&" already exists.  Do you want to write over it
(Y/N)?";
3540        INPUT "",A$
3550        IF UPC$(A$)="N" THEN 3490
3560        IF UPC$(A$)="Y" THEN 3590
3570        BEEP
3580        GOTO 3530
3590        PURGE File$
3600        CREATE File$,105,70
3610        ASSIGN #1 TO File$
3620        PRINT #1,1;Nopictures,Nofigures
3630        READ #1,2
3640        PRINT #1;Pictures(*),File$(*)
3650        BEEP
3660        DISP "PROGRAM COMPLETED"
3670        END


 *************************


3690 SUB Repro(#1,N,T,Pictures(*),P)
3700 !  #1 is the file pointer to the primitive file of the figure to
be drawn.
3710 !  N is the number of points in that file.
3720 !  T is which entity of the main picture is being drawn.
3730 !  Pictures(*) is the transformation array.
3740 !  P is the pen code (1 or 2 for draw, -1 for erase).
3750 OPTION BASE 1
3760 DIM X(N,3)     ! Allocate the necessary space for the points.
3770 READ #1,2      ! Set the serial pointer.
3780 READ #1;X(*)   ! Read in the points.
3790 GRAPHICS
3800 PEN P          ! Set the pen code
3810 MOVE Pictures(T,5),Pictures(T,6)    ! Move to real coordinates
3820 DEG
3830 PDIR Pictures(T,2)       ! Set rotation angle
3840 FOR I=1 TO N   ! Draw the figure, allowing for menu displacement
3850 RPLOT (X(I,1)-Pictures(T,3))*Pictures(T,7),(X(I,2)-Pictures(T,4))
*Pictures(T,7),X(I,3)
3860 NEXT I
3870 PENUP
3880 SUBEXIT
```

```
*************************


3900 SUB Menu(#1,J,N)
3910 OPTION BASE 1
3920 !  #1 is the file pointer.
3930 !  J is which menu item is being drawn.
3940 !  N is the number of points in the file.
3950 DIM X(N,3)          ! Allocate space for points
3960 READ #1,2           ! Set the serial pointer.
3970 READ #1;X(*)        ! Read in the points
3980 SETUU               ! Set UU's for clipping.
3990 LOCATE 10*(J-1),10*J,90,100  ! Pick place where figure goes in me
nu
4000 FRAME
4010 SCALE 0,1,0,1       ! Reproduce original scale
4020 FOR I=1 TO N        ! Draw the menu picture
4030 PLOT X(I,1),X(I,2),X(I,3)
4040 NEXT I
4050 SUBEXIT


*************************


4070 SUB Primitive(#1,W,N)
4080 !  #1 is the file pointer of the primitive to be drawn.
4090 !  W is which figure is being drawn.
4100 !  N is the number of points in the file.
4110 OPTION BASE 1
4120 DIM X(N,3)             ! Allocate the array space necessary.
4130 GCLEAR
4140 FRAME
4150 SCALE 0,1,0,1
4160 READ #1,2              ! Set the serial pointer.
4170 READ #1;X(*)           ! Read in the array.
4180 FOR I=1 TO N           ! Draw the figure
4190 PLOT X(I,1),X(I,2),X(I,3)
4200 NEXT I
4210 SUBEXIT


*************************
```

```
4230 SUB Seek(Pictures(*),X,Y,N,W)
4240 !  Pictures(*) is the transformation array.
4250 !  X and Y are the coordinates of the point for which the nearest
 figure
4260 !   is sought.
4270 !  N is the number of entities is the picture.
4280 !  W is the number of the entity which is closest.
4290 Mindis=9.9999999999E99      ! Set a dummy minimum distance value.
4300 W=0
4310 FOR I=1 TO N
4320 D=FNDis(I)     ! Find the distance to the ith entity of the pictur
e.
4330 IF D>=Mindis THEN Next    ! Check to see if the new distance is le
ss than
4340                           !  the previous minimum known distance.
4350 W=I                       ! Save the subscript of the new minimum
distance
4360 Mindis=D                  ! Save the new minimum distance.
4370 Next: NEXT I
4380 SUBEXIT
4390 DEF FNDis(I)=SQR((Pictures(I,5)-X)*(Pictures(I,5)-X)+(Pictures(I,
6)-Y)*(Pictures(I,6)-Y))
4400 SUBEND


 ************************


4420 SUB Movefigure(#1,N,W,Pictures(*))
4430 !  #1 is the file pointer of the figure to be moved.
4440 !  N is the number of points in that file.
4450 !  W is which entity in the picture is being moved.
4460 !  Pictures(*) is the transformation array.
4470 OPTION BASE 1
4480 DIM X(N,3)  ! Allocate the space needed.
4481 IF Pictures(W,1)=11 THEN 4510
4490 READ #1,2   ! Set the serial pointer.
4500 READ #1;X(*)! Read in the points of the figure.
4510         GOSUB Controlkeys         ! Enable the cusor control keys.
4520         ON KEY #11 GOTO Setfigure! Enable the set keys.
4530         Increment=100/455         ! Set the standard increment
4540         Upinc=Sideinc=0           ! Zero out the Up and Side incre
ments
4550         H1=Pictures(W,5)          ! Save old points in case of CAN
CEL
4560         H2=Pictures(W,6)
4570         ON KEY #12 GOTO Cancel    ! Enable the cancel key
```

```
4580          PEN 1
4590          POINTER Pictures(W,5),Pictures(W,6),2
4600          GRAPHICS
4610          PEN 1
4620          GOTO 4600                    ! Wait for command
4630 Setfigure: H1=Pictures(W,5)           ! Get rid of old points
4640            H2=Pictures(W,6)
4650 Cancel:    GOSUB Offkeyscontrol       ! Disable keys
4660            Pictures(W,5)=H1           ! Restore the old values.
4670            Pictures(W,6)=H2
4680            GCLEAR
4690            SUBEXIT
4700 Controlkeys: ON KEY #1 GOSUB Up  ! Enable the cursor control keys
.
4710              ON KEY #17 GOSUB Ups
4720              ON KEY #9 GOSUB Down
4730              ON KEY #25 GOSUB Downs
4740              ON KEY #0 GOSUB Left
4750              ON KEY #16 GOSUB Lefts
4760              ON KEY #2 GOSUB Right
4770              ON KEY #18 GOSUB Rights
4780              RETURN
4790 Offkeyscontrol: OFF KEY #1        ! Disable POINTER keys
4800              OFF KEY #1
4810              OFF KEY #2
4820              OFF KEY #9
4830              OFF KEY #10
4840              OFF KEY #17
4850              OFF KEY #18
4860              OFF KEY #25
4870              OFF KEY #26
4880              RETURN
4890 Up:     Upinc=Increment
4900         GOTO Cursor
4910 Ups:    Upinc=Increment/10
4920         GOTO Cursor
4930 Down:   Upinc=-Increment
4940         GOTO Cursor
4950 Downs:  Upinc=-Increment/10
4960         GOTO Cursor
4970 Left:   Sideinc=-Increment
4980         GOTO Cursor
4990 Lefts:  Sideinc=-Increment/10
5000         GOTO Cursor
5010 Right:  Sideinc=Increment
5020         GOTO Cursor
5030 Rights:Sideinc=Increment/10
```

```
5040         GOTO Cursor
5050 Cursor:     IF Pictures(W,1)<>11 THEN 5120    ! Check to see if the
 figure to
5060                                               ! to be moved is a li
ne.
5070             MOVE Pictures(W,3),Pictures(W,4)
5080             PEN -1                            ! Erase the line.
5090             DRAW Pictures(W,5),Pictures(W,6)
5100             PEN 1
5110             GOTO 5130
5120             CALL Drawfig(N,W,Pictures(*),-1,X(*)) ! Erase the figu
re.
5130             Pictures(W,5)=Pictures(W,5)+Sideinc ! Update main pict
ure
5140             Pictures(W,6)=Pictures(W,6)+Upinc   !  reference point
s.
5150             IF Pictures(W,1)<>11 THEN 5220
5160             Pictures(W,3)=Pictures(W,3)+Sideinc ! If the figure is
 a line,
5170             Pictures(W,4)=Pictures(W,4)+Upinc   !  update the othe
r endpoint
5180             MOVE Pictures(W,3),Pictures(W,4)    ! Draw in the new
line.
5190             DRAW Pictures(W,5),Pictures(W,6)
5200             Upinc=Sideinc=0                     ! Reset the increm
ents.
5210             GOTO 5240                           ! Wait for the nex
t command.
5220             Upinc=Sideinc=0                     ! Reset the increm
ents.
5230             CALL Drawfig(N,W,Pictures(*),2,X(*))! Redraw the figur
e in its
5240             POINTER Pictures(W,5),Pictures(W,6),2 ! new position.
5250             RETURN


 ************************


5270 SUB Drawfig(N,T,Pictures(*),P,X(*))
5280 !  N is the number of points in the array X(*)
5290 !  T is which entity in the main picture is being drawn.
5300 !  Pictures(*) is the transformation array.
5310 !  P is the pen code (1 or 2 for draw, -1 for erase).
5320 !  X(*) is the array holding the raw points for the primitive
5330 PEN P    ! Set the pen code
5340 GRAPHICS
```

```
5350 MOVE Pictures(T,5),Pictures(T,6)  ! Move to main picture coordina
tes.
5360 DEG                               ! Set degrees for rotation angl
e
5370 PDIR Pictures(T,2)                ! Set rotation angle
5380 FOR I=1 TO N                      ! Draw the figure with the prop
er
5390                                   !  transformations.
5400 RPLOT (X(I,1)-Pictures(T,3))*Pictures(T,7),(X(I,2)-Pictures(T,4))
*Pictures(T,7),X(I,3)
5410 NEXT I
5420 PENUP
5430 SUBEXIT
```

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary tape transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "REPRO:T15"
   b. Press: EXECUTE
3. Start the program:
   a. Press: RUN
4. When "Do you want to use an old picture configuration (Y/N)?" appears in the display area:
   a. If you want to use an old picture:
      1) Type: Y
      2) Press: CONT
      3) Go to step 5.

      or
   a. If you do not want to use an old picture:
      1) Type: N
      2) Press: CONT
      3) Go to step 6.
5. When "Please enter the file name of the old picture" appears in the display area:
   a. Type: The name of the file you wish to use
   b. Press: CONT
   c. Go to step 8.

   Note: If "File is not available" appears in the display area, go back to step 4.
6. When "How many figures do you want in your library?" appears in the display area:
   a. Enter: The number of "primitives" you want in your library or menu
   b. Press: CONT
7. Perform this step for each figure you want to use in your library.
   a. When "Please enter the file name for figure #i" appears in the display area:

1) Type: The file name of the ith menu figure

2) Press: CONT

b. Repeat step 7 for each menu figure.

c. Note: If "File is not available" appears in the display area, go back to step 7a.

8. The 9845 will go into graphics mode, and the menu will be drawn across the top of the screen. Also, if an old picture is being used, it will also be drawn.

9. At this point, the following special function keys are defined:

Key 8 - Pick a figure from the menu

Key 10 - Go into line drawing mode

Key 3 - Move a figure

Key 4 - Delete a figure

Key 5 - Draw the screen (with the menu)

Key 13 - Draw the screen (without the menu)

Key 24 - Modify the menu

Key 26 - Convert the picture on the screen to a "primitive" which can be made part of the menu

Key 27 - Load the construction program (which will reload this program upon completion)

Key 15 - Save the picture configuration

a. Press: One of the above special function keys

b. Go to one of the following steps:

1) If you pressed key 8, go to step 10

2) If you pressed key 10, go to step 16

3) If you pressed key 3, go to step 19

4) If you pressed key 4, go to step 22

5) If you pressed key 5, go to step 9

6) If you pressed key 13, go to step 9

7) If you pressed key 24, go to step 24

8) If you pressed key 26, go to step 29

9) If you pressed key 27, go to step 31

10) If you pressed key 15, go to step 32

10. (Pick a figure from the menu) A blinking cross will appear on the screen. Position the cross using the keys marked DISPLAY on the 9845 keyboard (left arrow, up arrow, right arrow, down arrow). When the cross is inside the square of the menu item you wish to pick, press CONT.
11. The figure which was selected in step 10 will be blown up to fill the whole screen and a set of crosshairs will appear. Using the DISPLAY keys on the 9845 keyboard, position the crosshairs to a point on the "primitive" which you want to use as a reference point.
    a. Press: CONT
12. Next, the main picture will be redrawn (without the menu). A set of crosshairs will appear. Position the crosshairs to where you want the reference point selected in step 11 to appear using the DISPLAY keys on the 9845 keyboard.
    a. Press: CONT
13. When "What angle of rotation? appears in the display area:
    a. Enter: The rotation angle (in degrees)
       Note: A positive rotation angle causes the "primitive" to be rotated in a counter-clockwise direction.
    b. Press: CONT
14. When "Scale (1-10)?" appears in the display area:
    a. Enter: The scale factor
       Note: A scale factor of 1 will cause the figure to appear in the same size as it is drawn in the menu.
    b. Press: CONT
15. Go to step 9.
16. (Line drawing mode) A set of crosshairs will appear on the screen. Using the DISPLAY keys on the 9845 keyboard, position the crosshairs to where you want the first endpoint of the line to appear.
    a. Press: CONT
17. The crosshairs will appear again. Position them (in similar manner to that used in step 16) to where you want the second

endpoint of the line.

a. Press: CONT

18. Go to step 9.

19. (Move a figure) A set of crosshairs will appear. Using the DISPLAY keys on the 9845 keyboard, position the crosshairs near the figure (or line) you want to move.

a. Press: Key 14 (Seek)

Note: The move operation may be cancelled at any time by pressing Key 12.

20. A blinking cross will appear and a line will be drawn to the reference point of the nearest figure.

a. If the indicated figure (or line) is indeed the one you intended to move:

1) Press: Key 6 (accept)

2) Go to step 21.

or

a. If the indicated figure (or line) is not the one you intended to move:

1) Press: Key 7 (reject)

2) Go to step 19.

21. The blinking cross will appear at the reference point of the indicated figure (or line). The figure may now be moved by pressing the following keys:

Key 0 - left
Key 1 - up
Key 2 - right
Key 3 - down

Finer increments may be obtained by shifting the given keys.

a. When the figure (or line) has been positioned where you want it:

1) Press: Key 11 (set the figure)

2) Go to step 9.

Note: For quick repetition of a keystroke, hold down on the REPEAT key at the same time as you are holding down on the special function key.

22. (Delete a figure) A set of crosshairs will appear. Using the DISPLAY keys on the 9845 keyboard, position the crosshairs

near the figure (or line) you want to delete.

a. Press: Key 14 (Seek)

Note: The delete operation may be cancelled at any time by pressing Key 12.

23. A blinking cross will appear and a line will be drawn to the reference point of the nearest figure (or line).

a. If the indicated figure (or line) is indeed the one you wish to delete:

1) Press: Key 6 (accept)

2) Go to step 9.

or

a. If the indicated figure (or line) is not the one you wish to delete:

1) Press: Key 7 (reject)

2) Go to step 22.

24. (Modify the library) When "Do you want to add figures to the library, or delete them (A/D)?" appears in the display area:

a. If you want to add figures to the "primitive" menu:

1) Type: A

2) Press: CONT

3) Go to step 25.

or

a. If you want to delete figures from the "primitive" menu:

1) Type: D

2) Press: CONT

3) Go to step 27.

25. When "Please enter the file name of figure #i" appears in the display area:

a. Type: The file name of the ith figure

b. Press: CONT

26. When "Do you want to add more figures to the library (Y/N)?" appears in the display area:

a. If you want to add more figures to the library:

1) Type: Y

2) Press: CONT

3) Go to step 25.

or

a. If you do not want to add more figures to the library:
    1) Type: N
    2) Press: CONT
    3) Go to step 9.

27. When "Please enter the number of the figure you wish to delete from the library" appears in the display area:
    a. Enter: The number of the menu item you wish to delete (Note: The leftmost item on the screen in graphics mode is 1, and the menu items are numbered sequentially to the right.)
    b. Press: CONT

28. When "Do you want to delete more figures (Y/N)?" appears in the display area:
    a. If you want to delete more figures:
        1) Type: Y
        2) Press: CONT
        3) Go to step 27.

        or

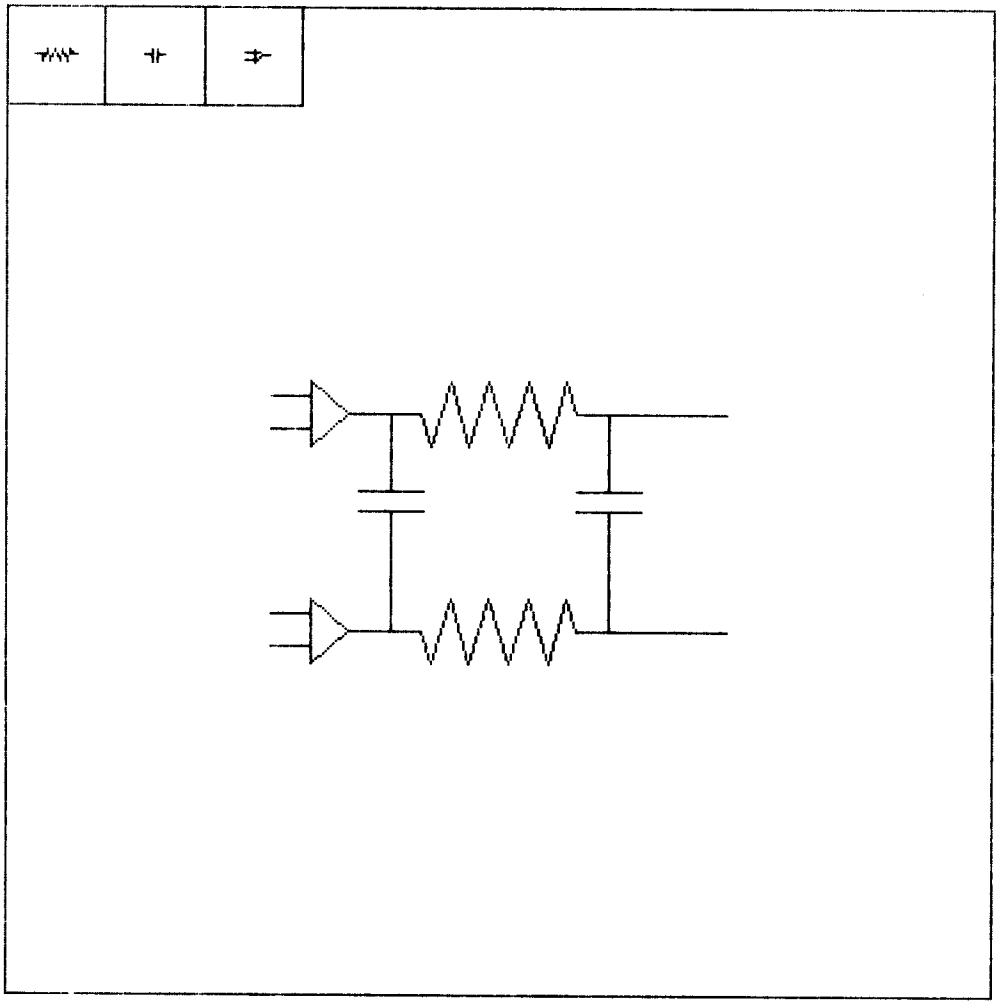
    a. If you do not want to delete more figures:
        1) Type: N
        2) Press: CONT
        3) Go to step 9.

29. (Convert to primitive) Wait for the conversion program to be loaded into memory. When "What do you want to call the new primitive file?" appears in the display area:
    a. Type: The name of the file you want to keep the converted figure in.
    b. Press: CONT

30. The entity reproduction program will be loaded back in. When the mass storage device is no longer busy, go back to step 9.

31. (Picture construction) The picture construction program will be loaded into memory. Follow the instructions given for that program. When you are done using that program, the entity reproduction program will be re-loaded. Go back to step 9 when the mass storage device is no longer busy.

32. (Save the picture configuration) When "Which file do you want to use to store the picture configuration?" appears in the display area:
    a. Type: The name of the file you wish to use to store the picture in.
    b. Press: CONT
33. The message "PROGRAM COMPLETED" will be printed and a beep will sound. At this point, the program stops.



HEWLETT hp PACKARD

# CONVERSION PROGRAM

This program takes an entity (created by the Entity Creation program in the file "REPRO) and converts it to a "primitive" which may be used as a menu item in the Entity Creation program.

This program is meant to be loaded from the Entity Creation program, but it can also be used as a stand-alone program.

Program Utilization:

File Name: CONVRT

Subprograms Required:

Transform
Ptransform
Ltransform

Variables (Main program):

| | | |
|---|---|---|
| Entrypoint | - | Tells whether the program is used as a stand-alone program, or if it is being loaded from the entity creation program |
| File$(*) | - | Array containing the names of the "primitive" files which contribute to the entity being converted |
| I | - | Loop counter |
| N | - | The number of points in each "primitive" file |
| Nofigures | - | The number of "primitives" available for the entity to be constructed from |
| Nopictures | - | The number of "primitives" which make up the entity being converted |
| Pictures(*) | - | The transformation array |
| Subscr | - | Tells how many points will go into the new "primitive" file |

Variables (Subprogram Transform):

| | | |
|---|---|---|
| A$ | - | Used for user's answer to (Y/N) question |
| C | - | Used with ASSIGN statement to see whether or not a file is available |
| Cscript | - | Current subscript of the last converted point |
| File$ | - | The name of the file where the converted entity is kept |
| File$(*) | - | The array holding the names of the "primitive" files which make up the entity being converted |
| I | - | Loop counter |
| J | - | Loop counter |
| N | - | The number of points in a "primitive" file |
| Nofigures | - | The number of "primitives" available |
| Nopictures | - | The number of "primitives" which make up the entity being converted |
| Pictures(*) | - | The transformation array |
| Points(*) | - | The array holding the converted points |
| Subscr | - | The total number of points in the converted entity |

Variables (Subprogram Ltransform):

| | | |
|---|---|---|
| Cscript | - | Current subscript of the last converted point |
| I | - | Loop counter |
| Pictures(*) | - | The transformation array |
| Points(*) | - | The array holding the converted points |
| X | - | Tells which "primitive" is being converted |

Variables (Subprogram Ptransform):

| | | |
|---|---|---|
| Cos | - | Temporary variable for the cosine of the rotation angle |

| | | |
|---|---|---|
| Cscript | - | Current subscript of the last converted point |
| I | - | Loop counter |
| J | - | Loop counter |
| N | - | The number of points in the "primitive" being converted |
| Pictures(*) | - | The transformation array |
| Points(*) | - | The array holding the converted points |
| Sin | - | Temporary variable for the sine of the rotation angle |
| T1 | - | Temporary variable for rotating the X coordinate |
| T2 | - | Temporary variable for rotating the Y coordinate |
| X | - | Tells which "primitive" is being converted |
| X(*) | - | The array containing the points being transformed and converted |

Special Considerations and Programming Hints:

1. This program always converts the information in file "!@#$%^". This is for easy communication with the Entity Creation program. Thus, you must rename the file you wish to convert to "!@#$%^" before running this program.
2. System Configuration:
   Standard Memory Option

# Listing of Conversion Program

```
10 Ep1:   Entrypoint=1  ! There are two entry points to this program.
 One is
20                      !  for a stand-alone run,
30   GOTO 60
40 Ep2:   Entrypoint=2  !  the other is for a link from the entity rep
roduction
50                      !  program.
60   OPTION BASE 1
70   DIM File$(10),Pictures(100,7)
80   ASSIGN #1 TO "!@#$%^"              ! Assign the file to be converted
90   READ #1,1;Nopictures,Nofigures     ! Read the number of primitives i
n the file
100  READ #1,2
110  READ #1;Pictures(*),File$(*)       ! Read in the transformation arra
y
120  FOR I=1 TO Nofigures
130  ASSIGN #I TO File$(I)              ! Assign the primitive files
140  NEXT I
150  Subscr=0        ! Subsr will tell how many points will be needed f
or
160                  ! the new primitive that's being built.
170  FOR I=1 TO Nopictures                  ! This loop finds the total
number
180  IF Pictures(I,1)<>11 THEN Primcheck  !  of points that wi   be co
nverted.
190  Subscr=Subscr+2
200  GOTO Nextcheck
210 Primcheck:  READ #Pictures(I,1),1;N
220  Subscr=Subscr+N
230 Nextcheck:  NEXT I
240  CALL Transform(Subscr,Pictures(*),File$(*),Nopictures,Nofigures)
250  ON Entrypoint GOTO 270,260
260  LOAD "REPRO",Ep3
270  END


**************************


290  SUB Transform(Subscr,Pictures(*),File$(*),Nopictures,Nofigures)
300  !  Subscr is the total number of points to be converted.
310  !  Pictures(*) is the transformation array.
320  !  File$(*) holds the names of the files of the primitives being
converted.
330  !  Nopictures is the number of entities to be transformed.
340  !  Nofigures is the number of primitives available to choose from
```

```
350  OPTION BASE 1
360  DIM Points(Subscr,3)       ! Allocate the array for the converted
points.
370  Cscript=0                  ! Current subscript
380  FOR I=1 TO Nofigures       ! Assign the files.
390  ASSIGN #I TO File$(I)
410  NEXT I
420  FOR I=1 TO Nopictures      ! This loop transforms the points by ca
lling
430                             !  two subprograms.
440  IF Pictures(I,1)<>11 THEN Primread   ! Check to see if the entity
 to be
450                                       !  converted is a line or a
figure.
460  CALL Ltransform(Cscript,Pictures(*),I,Points(*))
470  GOTO Nextread
480 Primread:  READ #Pictures(I,1),1;N
490  CALL Ptransform(#Pictures(I,1),Cscript,Pictures(*),N,I,Points(*))
500 Nextread:  NEXT I
510  INPUT "What do you want to call the new primitive file?",File$
520  ASSIGN #1 TO File$,C
530  IF C=1 THEN Create
540  BEEP
550  DISP File$&" already exists.  Do you want to overwrite it (Y/N)?"
;
560  INPUT "",A$
570  IF UPC$(A$)="N" THEN 510
580  IF UPC$(A$)="Y" THEN 610
590  BEEP
600  GOTO 550
610  PURGE File$
620 Create:  CREATE File$,Subscr+1,30
630  ASSIGN #1 TO File$
640  PRINT #1,1;Subscr
650  READ #1,2
660  FOR J=1 TO Subscr
670  PRINT #1,J+1;Points(J,1),Points(J,2),Points(J,3)
680  NEXT J
690  SUBEXIT


**************************


710  SUB Ltransform(Cscript,Pictures(*),X,Points(*))
720  !  Cscript is the subscript of the last point to have been conver
ted.
```

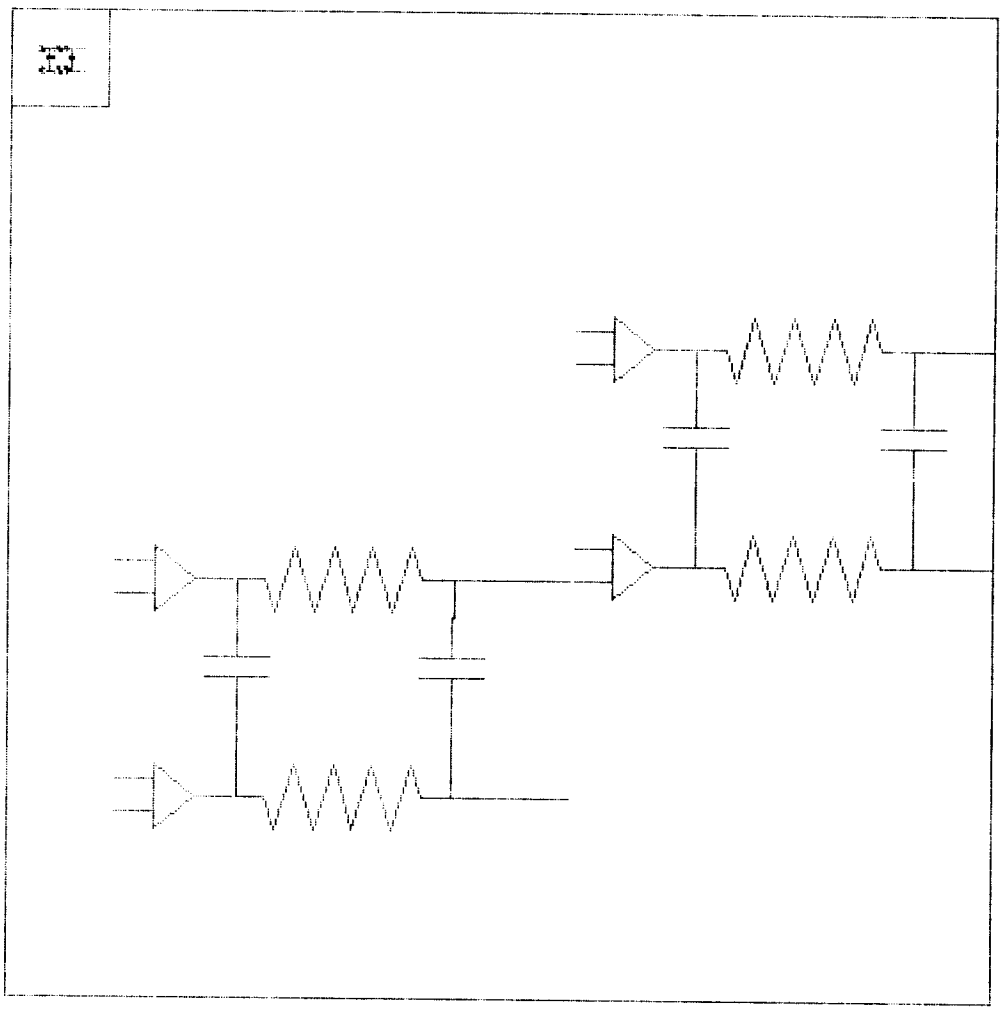
```
730   !  Pictures(*) is the transformation array.
740   !  X tells which entity is being converted.
750   !  Points(*) is the array which will hold the converted points.
760   FOR I=Cscript+1 TO Cscript+2
770                         ! The endpoints of the line must be normalized
to fall
780                         !  between 0 and 1.  The translation of 5 is be
cause the
790                         !  entity reproduction program scales from -5 t
o +5.  The
800                         !  same argument is the reason for the division
 by 10.
810   Points(I,1)=(Pictures(X,1+2*(I-Cscript))+5)/10
820   Points(I,2)=(Pictures(X,2+2*(I-Cscript))+5)/10
830   NEXT I
840   Points(Cscript+1,3)=-2
850   Points(Cscript+2,3)=-1
860   Cscript=Cscript+2 ! Update the current subscript
870   SUBEXIT


**************************


890   SUB Ptransform(#1,Cscript,Pictures(*),N,X,Points(*))
900   !  #1 is the file pointer of the figure being transformed.
910   !  Cscript is the subscript of the last point to have been conver
ted.
920   !  Pictures(*) is the transformation array.
930   !  N is the number of points in file #1.
940   !  X tells which entity is being transformed.
950   !  Points(*) is the array where the converted points go.
960   OPTION BASE 1
970   DIM X(N,3)     ! Allocate the array to hold the primitive's points

980   READ #1,2      ! Position the serial pointer.
990   READ #1;X(*)   ! Read in the data array of the figure to be transf
ormed
1000                !  and converted.
1010 FOR I=Cscript+1 TO Cscript+N     ! These loops translate and scale
 the
1020   FOR J=1 TO 2                   !  primitive to main picture coor
dinates.
1030    Points(I,J)=(X(I-Cscript,J)-Pictures(X,J+2))*Pictures(X,7)
1040   NEXT J
1050 NEXT I
1060 DEG     ! Set to degrees mode for rotation angle.
```

```
1070 FOR I=Cscript+1 TO Cscript+N
1080 Cos=COS(Pictures(X,2))
1090 Sin=SIN(Pictures(X,2))
1100 Points(I,3)=X(I-Cscript,3)                ! Keep the old pen codes
1110 T1=Points(I,1)*Cos-Points(I,2)*Sin        ! Rotate the X coordinate
s
1120 T2=Points(I,1)*Sin+Points(I,2)*Cos        ! Rotate the Y coordinate
s
1130 Points(I,1)=(T1+Pictures(X,5)+5)/10       ! Translate and normalize
 the
1140 Points(I,2)=(T2+Pictures(X,6)+5)/10       ! X and Y coordinates
1150 NEXT I
1160 Cscript=Cscript+N
1170 SUBEXIT
```

User Instructions:

1. Insert the Utility Library cartridge 2 into the primary tape transport (i.e., the transport above the special function keys).
2. Load the file:
   a. Type: LOAD "CONVRT:T15"
   b. Press. EXECUTE
3. Rename the file to be converted:
   a. Type: RENAME "*file*" TO "!@#$%^", where *file* is the name of the file you wish to convert.
   b. Press: EXECUTE
   c. If the message "ERROR 54" is displayed at the bottom of the screen, then the file "!@#$%^" already exists and must be destroyed:
      1) Type: PURGE "!@#$%^"
      2) Press: EXECUTE
      3) Repeat step 3a.
4. Start the program:
   a. Press: RUN
5. When "What do you want to call the new primitive file?" appears in the display area of the CRT:
   a. Type: A valid file name (a file name may be up to six characters long)
   b. Press: CONT
6. The program stops.

   Note: If you want to save the old file which was converted, it is still named "!@#$%^". You may rename it using the RENAME command in a manner similar to the procedure in step 3. (See the Operating and Programming Manual for details).



DUMP GRAPHICS TO SELECT CODE

This binary program adds a basic statement that allows the CRT graphics display to be dumped to a raster scan device such as the HP 2631G Printer.

User Instructions:

1. Load the Utility Library Cartridge 2 into the tape transport.

2. Load the binary program:
   a. Type: LOAD BIN "DGRAPH"
   b. Press: EXECUTE

3. The graphics display can be dumped to the device by use of the DUMP GRAPHICS statement with the addition of the select code specifier. The statement may be included in a Basic program or executed from the keyboard. The statement syntax is:

   DUMP GRAPHICS [ #<select code>[,<bus address>];
   ][<lower bound>,[<upper bound>]]

The default select code is the internal thermal printer.

For example:

```
DUMP GRAPHICS     ! Dump to the internal thermal printer
DUMP GRAPHICS #3  ! Dump to select code 3
DUMP GRAPHICS #5,! Dump the horizontal band starting at 3Ø on
    2;3Ø,6Ø       ! the Y-axis (in current units) and ending at
                  ! 6Ø on the Y-axis (in current units).
                  ! The display will be dumped to select code 5,
                  ! bus address 2
```

HEWLETT hp PACKARD

# SEARCH

This program will search a program file in string form ("SAVE"d, not "STORE"d) and print out all occurrences of the strings it is searching for. Up to 20 strings may be searched for at a time; each string must be 80 characters or less. When a string is found, that string and the line in which it occurred will be printed.

Program Utilization:
File Name: SEARCH

Variables Used:
A$ - Used to input answers to Y/N questions.
File$ - Name of the file to be searched.
I - Loop counter
J - Loop counter
K - Number of strings to be changed.
Pos - Position of comma in select code input.
Psc$ - String for select code input.
Psc1 - Select code of printer.
Psc2 - Bus address of printer (if HP-IB).
S$ - String that program lines are read into.
Str$(*) - Array of strings to be searched for.

Special Considerations and Programming Hints:

1. This program is meant to be a stand alone program rather than a subprogram.

2. System Configuration:
   Standard Memory
   Optional Printer

User Instructions:

1. Insert Utility Library cartridge 3 into tape transport.

2. Load the program:
   a. Type: GET "SEARCH:T15"
   b. Press: EXECUTE

3. Start the program:
   a. Press: RUN

4. When "NAME OF FILE TO BE SEARCHED" is displayed:
   a. Enter: The name of the program file you want searched. You may add a mass storage device specifier, if necessary.
   b. Press: CONTINUE

5. When "PRINTER SELECT CODE" is displayed:
   a. Enter: The select code of your printer, if you want a printed copy of the results. The default is 16 (the CRT).
   b. Press: CONTINUE

6. When "STRINGS TO BE SEARCHED FOR" is displayed:
   a. Enter: A string you want to be searched for. Remember, blanks (spaces) are significant in these strings. If you have finished entering strings, do not enter anything.
   b. Press: CONTINUE

   If you didn't enter anything before pressing CONTINUE, go on to step 7. Otherwise, repeat step 6 up to 20 times.

7. When "CHANGES (Y/N)?" is displayed:
   a. If you want to change any of the strings you entered in step 6:
      1) Enter: Y
      2) Press: CONTINUE
      3) Go to step 8
   b. If you do not want to change any of the strings you entered:
      1) Enter: N
      2) Press: CONTINUE
      3) Go to step 10

8. When "WHICH STRING DO YOU WANT TO CHANGE?" is displayed:
   a. Enter: The number of the string you want to change.
   b. Press: CONTINUE

9. When "NEW #X STRING TO BE SEARCHED FOR" is displayed:
   a. Enter: The new string
   b. Press: CONTINUE
   c. Go to step 7

10. The search will now begin. A record of all matches of any string you entered will be printed on the selected printer.

HEWLETT hp PACKARD

# SEARCH & REPLACE

This program will search a program file in string form ("SAVE"d, not "STORE"d) for occurrences of as many as 20 strings. When any of these strings is found, it will be replaced with a corresponding string you have specified. The file will be re-written on a new file, whose size and name you specify, with all the replacements made. A list of all replacements made will be printed on a selected printer.

Program Utilization:
File Name: SEARPL

Variables Used:

A$ - Used to input answers to Y/N questions.
File$ - Name of the file to be searched.
Filename$ - Name of the new file to be created.
I - Loop counter
J - Loop counter
K - Number of string to be changed.
Lrepl - Length of string replacing old string.
Lstr - Length of string to be replaced.
New$ - Temporary variable to store partially formed string.
Numrec - Number of records in file to be created.
Old - Position of previous match in the string.
Pos - Position of string to be replaced, and also position of comma in select code input.
Psc$ - String to input printer select code.
Psc1 - Select code of printer.
Psc2 - Bus address of printer (if HP-IB).
Recsize - Size of one record (in bytes) for file to be created.
Repl$(*) - Array of replacement strings. If Str$(I) is found, it is replaced with Repl$(I).

Restart - A flag set to 1 if the program has been restarted.
S$ - String that program lines are read into.
Str$(*) - Array of strings to be searched for.
Temp$ - Temporary variable to store partially formed string in.

User Instructions

1. Insert Utility Library cartridge 3 into tape transport.

2. Load the program:
   a. Type: GET "SEARPL"
   b. Press: EXECUTE

3. Start the program:
   a. Press: RUN

4. When "NAME OF FILE TO BE SEARCHED?" is displayed:
   a. Enter: The name of the program file you want searched. You may include a mass storage device specifier, if desired.
   b. Press: CONTINUE

5. When "NAME OF THE NEW FILE?" is displayed:
   a. Enter: The name of a file that the program will create to store the modified program.

6. When "RECORD SIZE OF THE NEW FILE?" is displayed:
   a. Enter: The record size (number of bytes per record). Refer to the partial catalog displayed at the top of the screen. The same number as that shown for the file to be searched is usually the best.
   b. Press: CONTINUE

7. When "NUMBER OF RECORDS IN NEW FILE?" is displayed:
   a. Enter: The number of records for the file to be created. This number would be equal to or slightly larger than the number of records in the file to be searched. If you are replacing short strings with long strings, you should make the new file slightly larger.
   b. Press: CONTINUE

At this point, "CREATING NEW FILE" should be displayed. If you entered a file name that already exists, "DUPLICATE FILE NAME -- TRY AGAIN" will be displayed. If this happens, go to step 5 and choose a different file name or a different mass storage device.

8. When "PRINTER SELECT CODE (DEFAULT = 16)" is displayed:
   a. Enter: The select code of the printer you wish to use. The default is 16 (the CRT).
   b. Press: CONTINUE

9. When "STRING TO BE SEARCHED FOR" is displayed:
   a. Enter: A string you want to have the program search for, 80 characters or less. You may include blanks (spaces) in this string, and they will be included in the string to be matched. If you have no more strings to enter, do not enter anything.
   b. Press: CONTINUE

   If you did not enter anything, go to step 11. Otherwise, go to step 10.

10. When "STRING TO REPLACE IT WITH" is displayed:
    a. Enter: The replacement string (80 characters or less). Blanks also are significant in this step.
    b. Press: CONTINUE

    You may enter up to 20 pairs of strings. If you have already entered 20 pairs at this point, go to step 11. Otherwise, go to step 9.

11. When "CHANGES? (Y/N)" is displayed:
    a. If you want to change a string you entered:
       1) Enter: Y
       2) Press: CONTINUE
       3) Go to step 12

b. If you do not want to change any of the strings:
   1) Enter: N
   2) Press: CONTINUE
   3) Go to step 15

12. When "WHICH NUMBER DO YOU WANT TO CHANGE?" is displayed:
   a. Enter: The number of the string you want to change.
   b. Press: CONTINUE

13. When "# X -- STRING TO BE SEARCHED FOR" is displayed:
   a. Enter: The new string for that number. If you do not want to change it, enter nothing.
   b. Press: CONTINUE

14. When "STRING TO REPLACE IT WITH" is displayed:
   a. Enter: The new replacement string. If you do not wish to change it, enter nothing.
   b. Press: CONTINUE
   c. Go to step 11

15. At this point, the search will start, and you will get a printout showing all replacements made. When the program is finished, "SEARCH COMPLETE" will be displayed. It is possible that the file that was created was too small. If this happens, the program will ask you if you want to re-create the file yourself or if you want the program to do it for you. If you do it yourself, you will have to start the program again from the beginning. If the program does it, it will use the same strings you had entered.

This program will read the contents of a data file and list them on a printer. It gives the type and value of each numeric data item, and the string and its length for a string data item. It can be very useful to determine the contents of a data file without having the program that uses that data file, and is also useful when debugging programs using a data file.

Program Utilization:
File Name: DUMP

Variables Used:
Main Program:
File - File number assigned to file being dumped.
File$ - Name of file being dumped.
Num - Number of records being dumped.
Pos - Position of comma in select code.
Psc$ - Printer sleect code string.
Psc1 - Printer select code.
Psc2 - Printer bus address (if HP-IB).
R - Return variable for ASSIGN statement.
Reclen - Record length (in bytes) of file being dumped.
Start - Record number where dump will start.
Work$ - String input variable for answers to questions.

Subroutine Set-up:
Buffer$ - Used too for a string data item for printing.
Bytes - Number of bytes remaining in current record.
I - Loop counter.
Len - Number of bytes in current data field.
Num - Number of records to dump.
Rec - Loop counter - current record number.
Reclen - Record length (in bytes) of file being dumped.

Start - Record number where dump will start.

String$ - Variable for reading string data items.

Val - Variable for reading numeric data items.

User Instructions:

1. Insert the Utility Library cartridge 3 into the tape transport.

2. Load the program:
   a. Type: ~~GET~~ LOAD "DUMP:T15"
   b. Press: EXECUTE

3. Run the program:
   a. Press: RUN

4. When "FILE NAME?" is displayed:
   a. Enter: The name of the file you want dumped. It must be a data file.
   b. Press: CONTINUE

   If the file you entered is protected, you will be asked to enter the protection code. Otherwise, go to step 6.

5. When "WHAT IS ITS PROTECTION CODE?" is displayed:
   a. Enter: The protection code.
   b. Press: CONTINUE

6. When "PRINTER SELECT CODE?" is displayed:
   a. Enter: The select code of the printer on which you want the output.
   b. Press: CONTINUE

7. When "RECORD LENGTH (in bytes)?" is displayed:
   a. Enter: The record length shown in the partial catalog displayed on the screen.
   b. Press: CONTINUE

8. When "BEGIN DUMP AT RECORD #?" is displayed:
   a. Enter: The number of the first record you want dumped.
   b. Press: CONTINUE

9. When "DUMP HOW MANY RECORDS?" is displayed:
   a. Enter: The number of records you want dumped.
   b. Press: CONTINUE

10. The dump will now begin. Several possible errors can be caught by the program, and an appropriate message will inform you of what was wrong.

This program will list programs in a variety of formats which allow you to set the number of columns per line and the number of lines per page. In addition to the variable format, you can list up to 20 programs at a time, without attending to the machine.

Program Utilization:
File Name: GLISTS

Variables Used:

A$ - Next program line read into this string. Also used for answers to Y/N questions.

Bottom_margin - Number of blank lines to print at bottom of page.

Colperline - Number of columns per line.

Count - Number of lines printed so far on the page.

File$(*) - Contains the names of the files to be listed.

Hpib - HP-IB printer bus address. Hpib = 999 if not using an HP-IB printer.

I - Loop counter.

K - Number of file being changed.

L - Length of program line being processed.

Lenline - Length of program line being processed.

Lineline - Number of lines that current program line will take to be printed.

Linesperpage - Number of lines printed on a page.

List - Loop counter, current file being processed.

Listings - Number of files to be listed.

Nolines - Number of lines to skip for $^{L}F$ characters.

Page - Number of lines that will fit on a page.

Perf - =1 if you are using perforated paper, =0 if you aren't.

Select - Printer select code.

Top_margin - Number of blank lines to be printed at top of page.

X - Position of "$^{L}$F" in program line.

Y - Position (greater than X) of "$^{L}$F" in program line.

User Instructions:

1. Insert the Utility Library cartridge 3 into the tape transport.

2. Load the file:
   a. Type: GET "GLIST3:T15"
   b. Press: EXECUTE

3. Start the program:
   a. Press: RUN

4. When "HOW MANY DILES DO YOU WANT TO LIST (MAX=20)" is displayed:
   a. Enter: The number of programs you want to list (up to 20).
   b. Press: CONTINUE

5. When "ENTER FILE NUMBER X WITH MASS STORAGE DEVICE IF NECESSARY" is displayed:
   a. Enter: The name of the file that the program is on, and a mass storage device specifier if necessary.
   b. Press: CONTINUE
   Repeat step 5 for each file.

6. When "CHANGES (Y/N)" is displayed:
   a. If you made an error in one of the file names:
      1) Enter: Y
      2) Press: CONTINUE
      3) Go to step 7
   b. If you do not wish to make any changes in the file names:
      1) Enter: N
      2) Press: CONTINUE
      3) Go to step 9

7. When "CHANGE WHICH NUMBER?" is displayed:
   a. Enter: The number of the file you want to change.
   b. Press: CONTINUE

8. When "NEW FILE # X" is displayed:
   a. Enter: The new file name.
   b. Press: CONTINUE
   c. Go to step 6

9. When "DO YOU HAVE AN HP-IB PRINTER" is displayed:
   a. If your printer uses an HP-IB interface:
      1) Enter: Y
      2) Press: CONTINUE
      3) Go to step 10
   b. If your printer does not use an HP-IB interface:
      1) Enter: N
      2) Press: CONTINUE
      3) Go to step 10

10. When "PRINTER SELECT CODE?" is displayed:
    a. Enter: The select code of the printer.
    b. Press: CONTINUE

    If you don't have an HP-IB printer, go to step 12.

11. When "BUS ADDRESS?" is displayed:
    a. Enter: The HP-IB bus address of your printer.
    b. Press: CONTINUE

12. When "DO YOU HAVE PERFORATED PAPER (Y/N)?" is displayed:
    a. If you have perforated paper:
       1) Enter: Y
       2) Press: CONTINUE
       3) Go to step 13

13. When "HOW MANY LINES WILL FIT ON A PAGE?" is displayed:
    a. Enter: The number of lines that would fit on the page if you filled it - not necessarily the number of lines you want printed on a page.
    b. Press: CONTINUE

14. When "HOW MANY LINES PER PAGE DO YOU WANT PRINTED?" is displayed:
    a. Enter: The number of lines you want printed on each page. If you specify less than the number that will fit, the page will be centered.
    b. Press: CONTINUE

15. When "HOW MANY COLUMNS PER LINE DO YOU WANT?" is displayed:
    a. Enter: The maximum number of characters you want printed on one line. Any longer lines will be printed on two (or more) lines.
    b. Press: CONTINUE

16. The listings will now be printed for all the files you entered. If a file you entered does not exist, the program will print a message to that effect and then continue with the next file.

HEWLETT hp PACKARD

# COMPARE

This program compares two program files to determine if they are exactly the same. If they are not, it will print the lines that do not compare exactly. The programs may be compared in two ways:

1) Matching line numbers; this means that only program lines with matching line numbers will be compared. Non-matching line numbers will be ignored.

2) Sequential comparison; this means that lines (including line numbers) are compared as they are encountered.

Program Utilization:
File Name: CMPARE

Variables Used:
A$ - Strings from program 1.
B$ - Strings from program 2.
Ended$ - Name of first file whose end-of-file mark is reached.
F1$ - Name of file that program 1 is on.
F2$ - Name of file that program 2 is on.
N1 - Line number of string from program 1.
N2 - Line number of string from program 2.
Pos - Position of comma in select code.
Psc$ - Select code input string.
Psc1 - Printer select code.
Psc2 - Printer bus address.
Q$ - Used to input answer to Y/N question.
Ra - Loop counter.
Rb - Loop counter.

S - Printer select code.

T$ - String used to check for normal termination if both files are of same length.

Special Considerations and Programming Hints:

1. This program is meant to be a stand alone problem solver rather than a subprogram.

2. System Configuration:
   Standard Memory Option
   (Optional) Printer

User Instructions:

1. Insert the Utility Library cartridge 3 into the tape transport.

2. Load the file:
   a. Type: GET "CMPARE:T15"
   b. Press: EXECUTE

3. Start the program:
   a. Press: RUN

4. When "ENTER THE NAME OF THE FILE THAT THE FIRST PROGRAM IS ON" is displayed:
   a. Enter: The file name, with mass storage device specifier, if necessary.
   b. Press: CONTINUE

5. When "ENTER THE NAME OF THE FILE THAT THE SECOND PROGRAM IS ON" is displayed:
   a. Enter: The file name, with mass storage device specifier, if necessary.
   b. Press: CONTINUE

6. When "ENTER THE PRINTER SELECT CODE (DEFAULT IS 16)" is displayed:
   a. Enter: The printer select code. The CRT is 16 (the default).
   b. Press: CONTINUE

7. When "COMPARING LINE NUMBERS ONLY (Y/N)?" is displayed:
   a. If you are comparing matching line numbers only:
      1) Enter: Y
      2) Press: CONTINUE
      3) Go to step 8
   b. If you want a sequential comparison:
      1) Enter: N
      2) Press: CONTINUE

3) Go to step 8

8. Mismatches will now be printed on the printer you selected. If one file was exhausted before the other, a message to that effect will be printed, giving the last two lines compared. Otherwise, "NORMAL TERMINATION" will be printed. In either case, the program will stop.

# FILE COPY

Introduction:

This program is a file handling utility. Files can be selectively copied, purged, protected and renamed.

Files on the source mass storage device can be completely copied to a destination mass storage device or selectively copied to a number of different devices.

The files that the user desires to copy must be 9845A files.

For a file to be successfully copied there must be:

1. No file on the "TO" device with the same name
2. Directory space available on the "TO" device
3. File space available on the "TO" device

User Instructions:

1. Make sure the power is on.
   a. Type: SCRATCH A
   b. Press: EXECUTE

2. Load the program into memory.
   a. Type: LOAD "FCOPY"
   b. Press: EXECUTE

3. Start the program:
   a. Press: RUN

4. The program will then ask for the source device. These are entered by first typing the device type (T for tape, F for floppy, etc.) followed by the select code and unit number (if applicable).

   Examples: T15
   F8
   F8, Ø
   F8, 1
   C12, 1, Ø

   (For details on setting these codes, refer to the Mass Storage Techniques Manual).

   To enter the code into the program, press CONT.

5. A prompt will then return asking the user whether or not to include protected files in the functions the software offers. If the answer is yes,

   Type: Y
   Press: CONT
   otherwise
   Press: CONT

6. The next prompt asks the user to specify a specific group of

files to be worked upon. A special file group is selected by comparing letters in the specifier to each file in the directory. If the characters in the specifier correspond directly to the first characters in the file name, then that file is part of the group.

```
Example:  FILES:  INPUT
                  INNUS
                  OUTPUT
                  ITEM
Group Specifier:  IN
         Result:  INPUT
                  INNUS
```

Of all the files available in the example, only those with the first two letters 'IN' were selected.

```
     ENTER:  File group specifier
     Press:  CONT
```

7. The program will now ask for the destination device. Follow the example in step 4 to choose the correct destination device.

8. All files on the source device are listed on the CRT in blocks of 80. The file being acted upon is that file with the inverse video cursor next to it. To position the cursor for selecting the appropriate file, use the 'UP', 'DOWN', 'LEFT', 'RIGHT' arrows. The 'HOME' key will position the cursor at the first file on the screen, pressing 'CONTROL HOME' will position the cursor at the last file on the screen.

9. The special function keys allow the user to select the desired function for the file currently designated by the cursor. The special function keys are displayed on the bottom of the CRT. There are five different sets. Pressing key #7 will select the next set of 5 sets of keys.

Pressing 'CONTROL', 'SHIFT' or 'CONTROL SHIFT' and key will select the previous set of function keys. Should the function of any key be unclear, pressing Key #6 and the specific key will provide information about that key.

10. Should the key display at the bottom of the CRT ever disappear when no prompts are present, press any Special Function key and the key display will reappear.

Special Considerations:

1. In order to insure data integrity, the user may wish to use the CHECK READ feature, which causes the machine to read everything that is written on a mass storage device to insure that the data was written correctly. To take advantage of this, perform the following instructions prior to running the program:

   a. Perform steps 1 and 2 of the User Instructions on the following page.
   b. Type: CHECK READ
   c. Press: EXECUTE
   d. Continue following the User Instructions starting at step 3.

   Upon completion of the program, it is usually wise to disable the CHECK READ feature, since it causes extra wear and tear on the tape cartridges (i.e., twice as much tape motion):

   a. Type: CHECK READ OFF
   b. EXECUTE

   Bear in mind when the CHECK READ feature is in effect, all mass storage operations will be much slower than otherwise.

9845A to 9845B Key File Conversion Binary Program

This binary program allows a 9845A KEY file to be loaded into a 9845B. After the 9845A KEY file has been loaded in the Special Function Keys are usable exactly as they were on the 9845A. A 9845B KEY file can then be created using the STORE KEY command.

User Instructions:

1. Load the Utility Library cartridge 3 into the tape transport.

2. Load the binary program:
   a. Type: LOAD BIN "LDK45A"
   b. Press: EXECUTE

3. Load the key file:

The statement to load the keyboard can be executed from the keyboard or be included in a Basic program. The statement syntax is:

LOAD KEY 9845A <file specifier>,[<language specifier>]

The file specifier is the same as for any 9845 file

"<file name>:[<mass storage unit specifier>]"

The language specifier is optional; the default is A for STANDARD ASCII

| LANGUAGE SPECIFIER | MEANING |
|---|---|
| A(default) | 9845A KEY FILE contains only STANDARD ASCII characters |
| F | 9845A KEY FILE contains FRENCH alternate characters |
| G | 9845A KEY FILE contains GERMAN alternate characters |
| K | 9845A KEY FILE contains KATAKANA alternate characters |
| S | 9845A KEY FILE contains SPANISH alternate characters |

It is up to the user to provide the correct language specifier in order to get the alternate characters he desires. For instance, if the 9845A KEY FILE contains the 3 character sequence "shift out" "shift in", and "G" was the language specifier, then this would be interpreted as a GERMAN uppercase umlaut O. If "F" was the language specifier, the same sequence would be interpreted as a FRENCH lowercase c cedilla. If the 9845A KEY FILE contains alternate characters of a language different from the given language specifier, these characters may appear as blanks or as alternate characters from the set specified by the given language specifier.

| KEY FILES CREATED ON any one of | ARE COMPATIBLE WITH any one of |
|---|---|
| 9845A/S, opt.8ØØ STD ASCII | 9845B/T STD ASCII |
| 9845A/S, opt.815 FRENCH | 9845B/T, opt. 81Ø FRENCH |
| 9845A/S, opt.825 SPANISH | 9845B/T, opt. 82Ø SPANISH |
| 9845A/S, opt.835 GERMAN | 9845B/T, opt. 85Ø SWEDISH-FINNISH |
| 9845A/T, opt.84Ø KATAKANA | 9845B/T, opt. 84Ø KATAKANA |

4. Store the file as a 9845B key file for future use:
   a. Type: STORE KEY <file specifier>
   b. Press: EXECUTE

HEWLETT hp PACKARD

# SCREEN ADDRESSING

This collection of subroutines allows you to use some of the screen addressing excape code routines without constructing the sequences yourself. There are three subroutines, Posit, Vertical and Center. Subroutine Posit will display a string on the screen, beginning at the row and column you specify. Subroutine Center will horizontally center a string about a row and column you choose. Subroutine Vertical prints a string vertically and centers it above a row and column.

It is important to understand the row and column numbering scheme when using these routines. The numbering starts at the upper left corner of the screen. The columns are numbered 1 through 80. The rows are numbered down from the top of the screen and are numbered 1 through 20. Thus, the center of the top line of the screen would be row 1, column 40.

The 9835A understands a Ø base system. That is, the rows are Ø to 19, the columns are Ø to 79. The subroutines adjust the input to this system for you.

The input parameters for each subroutine, will be in the order: row, column, string. Therefore, to center the string "MIDDLE" in the center of the screen, you could use CALL Center (10,40, "MIDDLE").

There is no driver for these subroutines. If you want to use them, you must like them onto your program. After you have your program loaded into memory:

1. Put the Utility cartridge 3 into the tape transport.

2. a. Type: LINK "POSIT", X where X is a line number greater than the last line number in your program. POSIT is a file containing all three subroutines.

   b. Press: EXECUTE

   The subroutines will now be available to the program in memory.

Subprogram Name: POSIT
This subprogram will print a string on the screen at the row and column specified.

Subprogram Utilization:
File Name: POSIT
Calling Syntax: CALL Posit (X,Y,Q$)

Input Parameters:
X - Row of screen to be accessed.
Y - Column of screen to be accessed.
Q$ - The string to be printed.

Local Variables:
A - To adjust row to system recognized Ø base (rather than subroutine 1 base).
B - To adjust column to Ø base.

Subprogram Name: Vertical
This subprogram will print a string vertically, centered about the row and column specified.

Subprogram Utilization:
File Name: POSIT
Calling Syntax: CALL Vertical (X,Y,Text$)

Input Parameters:

X - Row of screen to be centered on.

Y - Column of screen to be centered on.

Text$ - String to be printed.

Local Variables:

A - To adjust the row number to the system's Ø-based numbering system.

B - To adjust the column number to the system Ø-based system.

L - Row to start printing in.

Subprogram Name: Center

This subprogram will print a string centered horizontally about the row and column entered.

Subprogram Utilization:

File Name: POSIT

Calling Systax: CALL Center (X,Y,Text$)

Input Parameters:

X - Row of screen to be centered on

Y - Column of screen to be contered on.

Text$ - String to be printed.

Local Variables:

A - To adjust row number to system Ø-based numbering system.

B - To adjust column number to Ø-based system.

L - Column to start printing in.

HEWLETT hp PACKARD

# Block Lettering

Subprogram Name: Charac

This subprogram enables a programmer to use block lettering in printouts for a program. The program uses the character set of the 9835 keyboard to generate larger letters, 5 characters in height and 5 characters in width. At most, 13 large letters will fit on one line of the CRT. If a line has fewer than 13 letters, then it will be centered. All upper case letters and numbers are available.

Subprogram Utilization:

File Name: CHARS

Calling Syntax: CALL Charac (Fill$,M$)

Input Parameters:

Fill$ - A one character string holding the character that the letters are to be formed with.

M$ - The message string to be printed in the large letters.

Local Variables:

A(*) - Holds the data for forming the letters.

A$(*) - String array to hold the formed string for printing.

B$ - Temporary string variable.

Bin - Result of function FNB, binary value of I.

Bin$ - String containing characters corresponding to one's in Bin.

Fill$ - The fill character - used to form the letters.

I - Loop counter.

J - Loop counter.

M$ - The message string.

N$ - The message string.

Test$ - String containing the character set accepted.

Method:

The program utilizes binary conversions to accomplish the character formation. Each letter is represented in terms of 5 numbers (see diagram). The 5 numbers are the decimal equivalent of the binary number designated by characters to be printed. For example, to create an "X" in a 5 by 5 field, the squares with a * should be printed. Representing row 1 in binary (with 1's for *'s) would be $10001_2$ or $17_{10}$. Similarly, row 2 would be $01010_2$ or $10_{10}$. Row 3 would be $00100_2$ or $4_{10}$. Row 4 is the same as row 2 and row 5 is the same as row 1. So the DATA statement for the letter "X" would be: DATA 17, 10, 04, 10, 17. You can create your own characters in this manner. Just add a DATA statement for each character at the bottom of the list of DATA statements in the program. Then add the new character to the end of the Test$ string. You will have to add 1 to the dimensions of Test$ and A(*) for each character you add. If you create a block character that does not have a corresponding character on the keyboard, then you must pick one of the keyboard characters you want to replace it with. Then put that keyboard character on the end of Test$ (as above). For example, suppose you create a block character like Δ. Since this character is not on the keyboard, you choose the $ to access it with. The $ will go on the end of Test$, and you will put a $ in a string when you want Δ to be printed.

5 X 5 Character Field

| | | | | | |
|---|---|---|---|---|---|
| Row 1 | * | | | | * |
| Row 2 | | * | | * | |
| Row 3 | | | * | | |
| Row 4 | | * | | * | |
| Row 5 | * | | | | * |

User Instructions:

1. Insert the Utility Library cartridge 3 into the tape transport.

2. Load the file:
   a. Type: GET"CHARD:T15"
   b. Press: EXECUTE

3. Start the program:
   a. Press: RUN

4. When "FILL CHARACTER?" is displayed:
   a. Enter: One character with which you want the letters to be formed. If you enter ' (single quote), a filled-rectangle character will be used (character 127).
   b. Press: CONTINUE

5. When "MESSAGE? (13 CHARACTERS OR LESS)" is displayed:
   a. Enter: A message to be printed that is not more than 13 characters in length.
   b. Press: CONTINUE
   c. There is a moment's pause while the data is loaded, then the string will be printed.
   d. Repeat step 5 as often as you wish. When you are done, press STOP.