

EDITING BY COMPUTER.

APPENDICES C & D.


```
1  BILLER: PROCEDURE OPTIONS(MAIN);
2  /* SAMPLE EDIT NO 2, TO DEMONSTRATE FRED PRE-PROCESSOR. CHECKS
   AN INVOICE - PROCESSING SECTIONS NOT INCLUDED */ DECLARE (ITEM,CUSTOMER,UNI
   TQTY,DISCRATE,INSRATE) FIXED DECIMAL (5,0), CARD CHARACTER (80),
   (UNITVAL,DISCOUNT,INSURANCE,FREIGHT,
   COST,BILLED) FIXED DECIMAL (7,2), FRTRATE FIXED DECIMAL (5,2),ER
   RCODE FIXED BIN;
3  ON ENDFILE (SYSIN) GO TO ENDING;
4  PUT FILE (SYSPRINT) SKIP(60);
5  * READ CARD AND EDIT IT */ READ: GET FILE (
6  SYSIN) EDIT (CARD) (A(80)); GET STRING(CARD) ED
7  IT (CUSTOMER,ITEM,UNITQTY,UNITVAL,COST, DISCRATE,DISCOUNT,INSRATE,I
   NSURANCE,FREIGHT,BILLED) (2 F(5,0),F(4,0),2 F(6,2),F(2,0),F(6
   ,2),F(2,0),3 F(6,2)); CALL INVOICE;
8  GO TO READ;
9  /* CLOSE LISTINGS */ EN
10 DING: PUT FILE (SYSPRINT) EDIT (' END OF LISTING ***') (SKIP(60), A
   (20)); INVOICE:PROCEDURE
11 ;DECLARE Q_#DIAG(40) PACKED CHARAC
12 TER(32),(Q_#FATAL,Q_#QUERY) FIXED BINARY(1),(Q_#ADJUST,Q_#DIGCT) FIXED BINARY: D
13 ECLARE ( Q_#ITEM , Q_#CUSTOMER , Q_#UNITQTY , Q_#DISCRATE , Q_#INSRATE ) FIXED D
   ECIMAL ( 5 , 0 ) , ( Q_#UNITVAL , Q_#DISCOUNT , Q_#INSURANCE , Q_#FREIGHT , Q_#C
   OST , Q_#BILLED ) FIXED DECIMAL ( 7 , 2 ) , Q_#FRTRATE FIXED DECIMAL ( 5 , 2 ) ; Q
   N ENDPAGE (SYSPRINT) PUT FILE (SYSPRINT) EDIT(' *** I N V O I C E E
   D I T R U N ***
   ') (PAGE,A(133));Q_#FATAL,Q_#QUERY,Q_#ADJUST,Q_#DIGCT=0;Q_#
   ITEM=ITEM ;Q_#CUSTOMER=CUSTOMER ;Q_#UNITQTY=UNITQTY ;
   Q_#DISCRATE=DISCRATE ;Q_#INSRATE=INSRATE ;Q_#UNITVAL=UN
   ITVAL ;Q_#DISCOUNT=DISCOUNT ;Q_#INSURANCE=INSURANCE ;Q_#FREIGHT
   =FREIGHT ;Q_#COST=COST ;Q_#BILLED=BILLED ;Q_#F
   RTRATE=FRTRATE ; IF ( ITEM = 0 ) || ( ITEM > 3127 ) THEN DO; Q_#FATAL=1; IF
   Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='ITEM CODE OUT OF RANGE
   ' ; GO TO Q1#001; END; Q1#001: IF ( CUSTOMER = 0 ) || ( CUSTOMER > 63381 ) THEN DO;
   Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='CUSTOME
   R CODE OUT OF RANGE'; GO TO Q1#002; END; Q1#002: IF ( BILLED = COST + INSURANC
   E + FREIGHT - DISCOUNT ) THEN DO; GO TO Q1#003; END; IF ( ABS ( BILLED - COST -
   INSURANCE - FREIGHT + COST * DISCRATE / 100 ) < .02 ) THEN DO; Q_#ADJUST=Q_#ADJUST
   +1; DISCOUNT = COST + INSURANCE + FREIGHT - BILLED ; IF Q_#DIGCT=40 THEN Q_#DIG
   CT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='DISCOUNT RECALCULATED'; GO TO Q2#001; END; IF
   ( ABS ( BILLED - COST - COST * INSRATE / 100 - FREIGHT + DISCOUNT ) < .02 ) THEN
   DO; Q_#ADJUST=Q_#ADJUST+1; INSURANCE = BILLED - COST + DISCOUNT - FREIGHT ; IF Q_
   #DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='INSURANCE RECALCULATED';
   GO TO Q2#001; END; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DI
   AG(Q_#DIGCT)='BILLED COST DOES NOT AGREE'; Q1#003: Q2#001: IF ( COST = UNITQTY
   * UNITVAL ) THEN DO; GO TO Q1#004; END; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIG
   CT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='COST INVALID FOR QUANTITY'; Q1#004: IF ( DISC
   RATE = 0 ) || ( DISCRATE = 5 ) || ( DISCRATE = 12 ) || ( DISCRATE = 20 ) THEN DO; IF ( DISCOU
   NT < ( COST * DISCRATE / 100 ) - .02 ) || ( DISCOUNT > ( COST * DISCRATE / 100 ) +
   .02 ) THEN DO; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DI
   GCT)='DISCOUNT MISCALCULATED'; GO TO Q2#002; END; GO TO Q1#005; END; Q_#FATAL=
   1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='DISCOUNT RATE IN
   VALID'; Q1#005: Q2#002: IF ( INSRATE = 0 ) || ( INSRATE = 2 ) || ( INSRATE = 3 ) || ( INSR
   ATE = 5 ) || ( INSRATE = 7 ) THEN DO; IF ( INSURANCE < ( COST * INSRATE / 100 ) - .02
   ) || ( INSURANCE > ( COST * INSRATE / 100 ) + .02 ) THEN DO; Q_#FATAL=1; IF Q_#DIGCT
   =40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='INSURANCE MISCALCULATED'; GO T
   O Q2#003; END; GO TO Q1#006; END; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_
   #DIGCT+1; Q_#DIAG(Q_#DIGCT)='INSURANCE RATE INVALID'; Q1#006: Q2#003: IF ( FRE
```

LEVEL 2JAN67

OS/360 PL/I COMPILER (F)

PAGE 1
DATE 67.142

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

S,A,ST,SIZE=108000,SM=(001,080)

THE COMPLETE LIST OF OPTIONS USED DURING THIS COMPILATION IS--

EBCDIC
CHAR60
NOMACRO
SOURCE2
COMP
SOURCE
ATR
NOXREF
NOEXTREF
NOLIST
LOAD
NODECK
FLAGE
STMT
SIZE=108000
LINECNT=060
OPT=00
SORMGIN=(001,080)

123872

FRED PRE-PROCESSOR ***
LOGICAL CROSS-REFERENCE TABLE FOR THIS EDIT IS

PAGE 3

ITEM	****	ITEM						
CUSTOMER	****	CUSTOMER						
UNITQTY	****	UNITQTY	UNITVAL	COST				
DISCRATE	****	DISCRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED	
INSRATE	****	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED	
UNITVAL	****	UNITQTY	UNITVAL	COST				
DISCOUNT	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
INSURANCE	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
FREIGHT	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
		FRTRATE						
COST	****	UNITQTY	DISCRATE	INSRATE	UNITVAL	DISCOUNT	INSURANCE	FREIGHT
		COST	BILLED	FRTRATE				
BILLED	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
FRTRATE	****	FREIGHT	COST	FRTRATE				

123868

FRED PRE-PROCESSOR ***

PAGE 4

END BILLER;

END OF THIS COMPILATION, NO ERRORS WERE NOTED IN THE EDITING STATEMENTS

FRED PRE-PROCESSOR ***
ERROR 'COST INVALID FOR QUANTITY'

PAGE 2

```
/* CHECK DISCOUNT RATE AND DISCOUNT */
IF DISCRATE=0,=5,=12,=20 THEN
  IF DISCOUNT /* ALLOW FOR ROUNDING */
  C <((COST*DISCRATE/100)-.02,
  C >((COST*DISCRATE/100)+.02) THEN
    ERROR 'DISCOUNT MISCALCULATED'
  ELSE
    **
```

```
ELSE
  ERROR 'DISCOUNT RATE INVALID'
```

```
/* CHECK INSURANCE RATE AND INSURANCE */
IF INSRATE=0,=2,=3,=5,=7 THEN
  IF INSURANCE /* ALLOW FOR ROUNDING */
  C <((COST*INSRATE/100)-.02,
  C >((COST*INSRATE/100)+.02) THEN
    ERROR 'INSURANCE MISCALCULATED'
  ELSE
    **
```

```
ELSE
  ERROR 'INSURANCE RATE INVALID'
```

```
/* CHECK FREIGHT, COMPUTE % FREIGHTAGE */
IF FREIGHT=0 THEN
```

```
  **
  COST/FREIGHT<10 THEN
    QUERY 'FREIGHT HIGH'
  ELSE
```

```
  **
  IF COST=0 THEN
    DEFINE FRTRATE=0
  ELSE
    DEFINE FRTRATE=FREIGHT/COST
  END EDIT INVOICE
```

123867

RESULTS FOR 29 ACCEPTED RECORDS ARE
AVERAGE MAXIMUM TEMP 75.290
AVERAGE MINIMUM TEMP 47.500
TOTAL RAINFALL 9.220
AVERAGE HOURS SUNSHINE 9.060

26432

```
1 FRED PRE-PROCESSOR ***
2
3 BILLER:  PROCEDURE OPTIONS(MAIN);
4          /* SAMPLE EDIT NO 2, TO DEMONSTRATE FRED PRE-PROCESSOR.
5           CHECKS AN INVOICE - PROCESSING SECTIONS NOT INCLUDED */
6 DECLARE  (ITEM,CUSTOMER,UNITQTY,DISCRATE,INSRATE) FIXED DECIMAL (5,0),
7          CARD CHARACTER (80),
8          (UNITVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) FIXED
9          DECIMAL (7,2), FRTRATE FIXED DECIMAL (5,2),ERRCODE FIXED BIN;
10
11         ON ENDFILE (SYSIN) GO TO ENDING;
12         PUT FILE (SYSPRINT) SKIP(60);
13         /* READ CARD AND EDIT IT */
14 READ:    GET FILE (SYSIN) EDIT (CARD) (A(80));
15         GET STRING(CARD) EDIT (CUSTOMER,ITEM,UNITQTY,UNITVAL,COST,
16         DISCRATE,DISCOUNT,INSRATE,INSURANCE,FREIGHT,BILLED)
17         (2 F(5,0),F(4,0),2 F(6,2),F(2,0),F(6,2),F(2,0),3 F(6,2));
18         CALL INVOICE;
19         GO TO READ;
20         /* CLOSE LISTINGS */
21 ENDING:  PUT FILE (SYSPRINT) EDIT (' END OF LISTING ***') (SKIP(60),
22         A(20));
23 START EDIT INVOICE
24         /* CODED AS INTERNAL PROCEDURE */
25         EDIT (ITEM,CUSTOMER,UNITQTY,DISCRATE,INSRATE) FIXED DECIMAL (5,0),
26         C(UNITVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) FIXED DECIMAL
27         C(7,2),FRTRATE FIXED DECIMAL (5,2)
28         LIST CUSTOMER,ITEM,UNITQTY,UNITVAL,COST,DISCRATE,DISCOUNT,
29         CINSRATE,INSURANCE,FREIGHT,FRTRATE,BILLED
30         HEADING '          *** I N V O I C E   E D I T   R U N ***'
31         /* END OF DECLARATIVE STATEMENTS */
32
33         /* FIRST CHECK ITEM CODE AND CUSTOMER CODE FOR VALIDITY */
34         IF ITEM=0,>3127 THEN
35         *** THE DECLARATIVE SECTION MUST CONTAIN ONE EDIT, ONE FLAG AND ONE LIST OR PRINTNONE STATEMENT
36         ERROR 'ITEM CODE OUT OF RANGE'
37         ELSE
38         **
39         IF CUSTOMER=0,>63381 THEN
40         ERROR 'CUSTOMER CODE OUT OF RANGE'
41         ELSE
42         **
43         /* CHECK BILLED AMOUNT = COMPONENT SUM */
44         IF BILLED=COST+INSURANCE+FREIGHT-DISCOUNT THEN
45         **
46         ELSE
47         IF /* TRY CALCULATED DISCOUNT OR INSURANCE */
48         C ABS(BILLED-COST-INSURANCE-FREIGHT+COST*DISCRATE/100)<.02 THEN
49         ADJUST DISCNT=COST+INSURANCE+FREIGHT-BILLED,
50         *** ADJUSTMENT IS TO INVALID ITEM
51         C 'DISCOUNT RECALCULATED'
52         ABS(BILLED-COST-COST*INSRATE/100-FREIGHT+DISCOUNT)<.02 THEN
53         ADJUST INSURANCE=BILLED-COST+DISCOUNT-FREIGHT,
54         C 'INSURANCE RECALCULATED'
55         ELSE
56         ERROR 'BILLED COST DOES NOT AGREE'
57
58         /* CHECK COST */
59         IF COST=UNITQTY*UNITVAL THEN
60         **
```


ELSE

ERROR 'COST INVALID FOR QUANTITY'

/* CHECK DISCOUNT RATE AND DISCOUNT */

IF DISCRATE=0,=5,=12,=20 THEN

IF DISCOUNT /* ALLOW FOR ROUNDING */

C <((COST*DISCRATE/100)-.02,

C >((COST*DISCRATE/100)+.02 THEN

ERROR 'DISCOUNT MISCALCULATED'

ELSE

**

ELSE

ERROR 'DISCOUNT RATE INVALID'

/* CHECK INSURANCE RATE AND INSURANCE */

IF INSRATE=0,=2,=3,=5,=7 THEN

IF INSURANCE /* ALLOW FOR ROUNDING */

C <((COST*INSRATE/100)-.02,

C >((COST*INSRATE/100)+.02 THEN

ERROR *INSURANCE MISCALCULATED*

*** AN ERROR MESSAGE MUST BE GIVEN AS LISTING OPTION WAS SELECTED

ELSE

**

ELSE

ERROR 'INSURANCE RATE INVALID'

/* CHECK FREIGHT, COMPUTE % FREIGHTAGE */

IF FREIGHT=0 THEN

**

COST/FREIGHT<10 THEN

QUERY 'FREIGHT HIGH'

ELSE

**

IF COST=0 THEN

DEFINE FRTRATE=0

*** LHS OF ASSIGNMENT ILLEGAL

ELSE

DEFINE FRTRATE=FREIGHT/COST

END EDIT INVOICE

112267

FRED PRE-PROCESSOR ***
LOGICAL CROSS-REFERENCE TABLE FOR THIS EDIT IS

ITEM	****	ITEM						
CUSTOMER	****	CUSTOMER						
UNITQTY	****	UNITQTY	UNITVAL	COST				
DISCRATE	****	DISCRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED	
INSRATE	****	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED	
UNITVAL	****	UNITQTY	UNITVAL	COST				
DISCOUNT	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
INSURANCE	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
FREIGHT	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
		FRTRATE						
COST	****	UNITQTY	DISCRATE	INSRATE	UNITVAL	DISCOUNT	INSURANCE	FREIGHT
		COST	BILLED	FRTRATE				
BILLED	****	DISCRATE	INSRATE	DISCOUNT	INSURANCE	FREIGHT	COST	BILLED
FRTRATE	****	FREIGHT	COST	FRTRATE				

FRED PRE-PROCESSOR ***

PAGE 4

END BILLER;

END OF THIS COMPILATION. ERRORS WERE NOTED AS SIGNALLED ABOVE.
DO NOT ATTEMPT TO RUN THIS PROGRAM UNTIL THESE ERRORS HAVE BEEN REMOVED.

```
BILLER:  PROCEDURE OPTIONS(MAIN);
          /* SAMPLE EDIT NO 2, TO DEMONSTRATE FRED PRE-PROCESSOR.
           CHECKS AN INVOICE - PROCESSING SECTIONS NOT INCLUDED */
DECLARE  (ITEM,CUSTOMER,UNITQTY,DISCRATE,INSRATE) FIXED DECIMAL (5,0),
          CARD CHARACTER (80),
          (UNITVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) FIXED
          DECIMAL (7,2), FRTRATE FIXED DECIMAL (5,2),ERRCODE FIXED BIN;

          ON ENDFILE (SYSIN) GO TO ENDING;
          PUT FILE (SYSPRINT) SKIP(60);
          /* READ CARD AND EDIT IT */
READ:    GET FILE (SYSIN) EDIT (CARD) (A(80));
          GET STRING(CARD) EDIT (CUSTOMER,ITEM,UNITQTY,UNITVAL,COST,
          DISCRATE,DISCOUNT,INSRATE,INSURANCE,FREIGHT,BILLED)
          (2 F(5,0),F(4,0),2 F(6,2),F(2,0),F(6,2),F(2,0),3 F(6,2));
          CALL INVOICE;
          GO TO READ;
          /* CLOSE LISTINGS */
ENDING:  PUT FILE (SYSPRINT) EDIT (' END OF LISTING ***') (SKIP(60),
          A(20));
          START EDIT INVOICE
          /* CODED AS INTERNAL PROCEDURE */
          EDIT (ITEM,CUSTOMER,UNITQTY,DISCRATE,INSRATE) FIXED DECIMAL (5,0),
          C(UNITVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) FIXED DECIMAL
          C(7,2),FRTRATE FIXED DECIMAL (5,2)
          LIST CUSTOMER,ITEM,UNITQTY,UNITVAL,COST,DISCRATE,DISCOUNT,
          CINSRATE,INSURANCE,FREIGHT,FRTRATE,BILLED
          HEADING '          *** I N V O I C E   E D I T   R U N   ***'
          FLAG ERRCODE
          /* END OF DECLARATIVE STATEMENTS */

          /* FIRST CHECK ITEM CODE AND CUSTOMER CODE FOR VALIDITY */
          IF ITEM=0,>3127 THEN
            ERROR 'ITEM CODE OUT OF RANGE'
          ELSE
            **
          IF CUSTOMER=0,>63381 THEN
            ERROR 'CUSTOMER CODE OUT OF RANGE'
          ELSE
            **
          /* CHECK BILLED AMOUNT = COMPONENT SUM */
          IF BILLED=COST+INSURANCE+FREIGHT-DISCOUNT THEN
            **
          ELSE
            IF /* TRY CALCULATED DISCOUNT OR INSURANCE */
            C ABS(BILLED-COST-INSURANCE-FREIGHT+COST*DISCRATE/100)<.02 THEN
              ADJUST DISCOUNT=COST+INSURANCE+FREIGHT-BILLED,
            C 'DISCOUNT RECALCULATED'
            ABS(BILLED-COST-COST*INSRATE/100-FREIGHT+DISCOUNT)<.02 THEN
              ADJUST INSURANCE=BILLED-COST+DISCOUNT-FREIGHT,
            C 'INSURANCE RECALCULATED'
            ELSE
              ERROR 'BILLED COST DOES NOT AGREE'

          /* CHECK COST */
          IF COST=UNITQTY*UNITVAL THEN
            **
          ELSE
```


DCL NO.	IDENTIFIER	ATTRIBUTES
219	Q2#002	STATEMENT LABEL CONSTANT
365	Q2#003	STATEMENT LABEL CONSTANT
22	Q_#ADJUST	AUTOMATIC,BINARY,FIXED(15,0)
416	Q_#COMPLT	STATEMENT LABEL CONSTANT
23	Q_#DAY	AUTOMATIC,DECIMAL,FIXED(3,0)
22	Q_#DIAG	(*),AUTOMATIC,PACKED,STRING,CHARACTER
22	Q_#DIGCT	AUTOMATIC,BINARY,FIXED(15,0)
23	Q_#FALL	AUTOMATIC,DECIMAL,FIXED(7,2)
22	Q_#FATAL	AUTOMATIC,BINARY,FIXED(1,0)
24	Q_#FORM	STATEMENT LABEL CONSTANT
23	Q_#HRS	AUTOMATIC,DECIMAL,FIXED(3,0)
136	Q_#L001	STATEMENT LABEL CONSTANT
203	Q_#L002	STATEMENT LABEL CONSTANT
282	Q_#L003	STATEMENT LABEL CONSTANT
349	Q_#L004	STATEMENT LABEL CONSTANT
23	Q_#MAX	AUTOMATIC,DECIMAL,FIXED(7,2)
23	Q_#MIN	AUTOMATIC,DECIMAL,FIXED(7,2)
23	Q_#MONTH	AUTOMATIC,STRING,CHARACTER
22	Q_#QUERY	AUTOMATIC,BINARY,FIXED(1,0)
6	READCARD	STATEMENT LABEL CONSTANT
	SYSIN	FILE,EXTERNAL
	SYSPRINT	FILE,EXTERNAL
2	TALLY	AUTOMATIC,INITIAL,DECIMAL,FIXED(3,0)
61	TST2	STATEMENT LABEL CONSTANT
1	WEATHER	ENTRY,DECIMAL,FLOAT(SINGLE)

WEATHER: PROCEDURE OPTIONS (MAIN);

PAGE 7

COMPILER DIAGNOSTIC MESSAGES

THE SUPPRESSION OF WARNING MESSAGES HAS BEEN REQUESTED.

TIME FOR THIS COMPILATION WAS 1.48 MINUTES

26438

1
2
3 RECORD IN ERROR
4 2FEB 83.10 61.10 0.00 21
5 HOURS TOO HIGH

6 RECORD IN ERROR
7 0FEB 84.10 54.02 0.00 8
8 DAY INVALID

9 RECORD IN ERROR
10 9FEB 110.04 72.04 0.00 12
11 MAXIMUM TOO HIGH

12 RECORD IN ERROR
13 10FEB 67.04 68.13 0.00 10
14 MIN > MAX

15 RECORD IN ERROR
16 11FAB 99.08 55.10 0.00 7
17 MONTH INVALID

18 RECORD IN ERROR
19 31FEB 92.01 57.37 0.00 10
20 DAY INVALID

21 RECORD QUERIED
22 17FEB 77.77 30.01 6.31 14
23 MINIMUM LOW
24 FALL HIGH

25 RECORD QUERIED
26 18FEB 76.41 34.12 0.00 12
27 MINIMUM LOW

28 RECORD QUERIED
29 21FEB 101.00 54.10 0.12 13
30 MAXIMUM HIGH

31 RECORD IN ERROR
32 24FEB 1.00 54.10 0.00 10
33 MIN > MAX

34 RECORD ADJUSTED
35 27FEB 80.70 63.27 0.00 4
36 MAX TEMP NOT STATED

37 RECORD IN ERROR
38 3AUG 78.02 28.03 0.00 7
39 MAXIMUM TOO HIGH

40 RECORD QUERIED
41 4AUG 73.10 27.02 0.00 9
42 MAXIMUM HIGH

43 RECORD IN ERROR
44 6ARG 0.00 0.00 0.00 0
45 MONTH INVALID
46 MAX TEMP NOT STATED
47 MIN TEMP NOT STATED
48
49
50

26493

ATTRIBUTE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES
2	CARD	AUTOMATIC,STRING,CHARACTER
17	COMPLETE	STATEMENT LABEL CONSTANT
2	DAY	AUTOMATIC,INITIAL,DECIMAL,FIXED(3,0)
25	DAYLIM	AUTOMATIC,DECIMAL,FIXED(3,0)
21	EDITOR	ENTRY,DECIMAL,FLOAT(SINGLE)
2	FALL	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
2	FALLTOT	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
2	FLAG	AUTOMATIC,BINARY,FIXED(15,0)
2	HRS	AUTOMATIC,INITIAL,DECIMAL,FIXED(3,0)
2	HRSTOT	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
	IQ_#	AUTOMATIC,BINARY,FIXED(15,0)
2	MAX	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
25	MAXLIM	AUTOMATIC,DECIMAL,FIXED(7,2)
2	MAXTOT	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
2	MIN	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
25	MINLIM	AUTOMATIC,DECIMAL,FIXED(7,2)
2	MINTOT	AUTOMATIC,INITIAL,DECIMAL,FIXED(7,2)
2	MONTH	AUTOMATIC,STRING,CHARACTER
53	Q1#001	STATEMENT LABEL CONSTANT
61	Q1#002	STATEMENT LABEL CONSTANT
73	Q1#003	STATEMENT LABEL CONSTANT
219	Q1#004	STATEMENT LABEL CONSTANT
365	Q1#005	STATEMENT LABEL CONSTANT
381	Q1#006	STATEMENT LABEL CONSTANT
389	Q1#007	STATEMENT LABEL CONSTANT
73	Q2#001	STATEMENT LABEL CONSTANT

FRED PRE-PROCESSOR ***

PAGE 3

LOGICAL CROSS-REFERENCE TABLE FOR THIS EDIT IS

MAX	****	MAX	MIN	MONTH
MIN	****	MAX	MIN	MONTH
FALL	****	FALL		
DAY	****	DAY		
HRS	****	HRS		
MONTH	****	MAX	MIN	MONTH

26448

FRED PRE-PROCESSOR ***

PAGE 4

END WEATHER;

END OF THIS COMPILATION, NO ERRORS WERE NOTED IN THE EDITING STATEMENTS

26447

LEVEL 2JAN67

OS/360 PL/I COMPILER (F)

PAGE 1
DATE 67.127

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

S,A,ST,SIZE=108000,SM=(001,080)

THE COMPLETE LIST OF OPTIONS USED DURING THIS COMPILATION IS--

EBCDIC
CHAR60
NOMACRO
SOURCE2
COMP
SOURCE
ATR
NOXREF
NOEXTREF
NOLIST
LOAD
NODECK
FLAG
STMT
SIZE=108000
LINECNT=060
OPT=00
SORMGIN=(001,080)

26444

```
1 WEATHER: PROCEDURE OPTIONS (MAIN);
2 /* FIRST SAMPLE EDIT TO DEMONSTRATE FRED PRE-PROCESSOR
3 EDITS TEMPERATURE AND RAINFALL CARDS */
4
5 1 DECLARE (MAX,MAXTOT,MIN,MINTOT,FALL,FALLTOT,HRSTOT) FIXED DECIMAL(7,2)
6 INITIAL (0),
7 (DAY,HRS,TALLY) FIXED DECIMAL (3,0) INITIAL (0),
8 CARD CHARACTER(80),
9 MONTH CHARACTER (3), FLAG FIXED BINARY;
10
11 3 ON ENDFILE (SYSIN) GO TO COMPLETE;
12 5 PUT FILE (SYSPRINT) SKIP(60);
13
14 /* READ NEXT CARD AND SUBMIT TO EDIT */
15 READCARD: GET FILE (SYSIN) EDIT (CARD) (A(80));
16 GET STRING (CARD) EDIT (DAY,MONTH,MAX,MIN,FALL,HRS)
17 (F(2),A(3),3 F(5,2),F(2));
18 CALL EDITOR;
19 IF FLAG=3 THEN GO TO READCARD; /* DO NOT USE ERRORS */
20 TALLY=TALLY+1; MAXTOT=MAXTOT+MAX; MINTOT=MINTOT+MIN;
21 FALLTOT=FALLTOT+FALL; HRSTOT=HRSTOT+HRS;
22 GO TO READCARD;
23
24 /* ALL CARDS READ, COMPUTE AND PRINT AVERAGES */
25 COMPLETE: MAXTOT=MAXTOT/TALLY; MINTOT=MINTOT/TALLY;
26 HRSTOT=HRSTOT/TALLY;
27 PUT FILE (SYSPRINT) EDIT ('RESULTS FOR ',TALLY,' ACCEPTED',
28 ' RECORDS ARE', 'AVERAGE MAXIMUM TEMP',MAXTOT,
29 'AVERAGE MINIMUM TEMP',MINTOT,'TOTAL RAINFALL',FALLTOT,
30 'AVERAGE HOURS SUNSHINE',HRSTOT)
31 (PAGE,X(2),A(12),F(4),A(9),A(12),SKIP,A(20),X(5),F(7,3),
32 SKIP,A(20),X(5),F(7,3),SKIP,A(14),X(11),F(7,3),SKIP,
33 A(22),X(3),F(7,3));
34
35 /* INCLUDE EDIT AS INTERNAL PROCEDURE */
36 EDITOR :PROCEDURE ;DECLARE Q_#D
37 IAG(40) PACKED CHARACTER(32),(Q_#FATAL,Q_#QUERY) FIXED BINARY(1),(Q_#ADJUST,Q_#D
38 IGCT) FIXED BINARY; DECLARE ( Q_#MAX , Q_#MIN , Q_#FALL ) FIXED DECIMAL ( 7 , 2
39 ) , ( Q_#DAY , Q_#HRS ) FIXED DECIMAL ( 3 , 0 ) , Q_#MONTH CHARACTER ( 3 ); Q_#F
40 ORM: FORMAT ( SKIP , F ( 2 ) , A ( 3 ) , 3 F ( 8 , 2 ) , F ( 4 ) ); DECLARE ( M
41 AXLIM , MINLIM ) FIXED DECIMAL ( 7 , 2 ) , DAYLIM FIXED DECIMAL ( 3 , 0 ) ;Q_#FA
42 TAL,Q_#QUERY,Q_#ADJUST,Q_#DIGCT=0;Q_#MAX =MAX ;Q_#MIN =M
43 IN ;Q_#FALL =FALL ;Q_#DAY =DAY ;Q_#HRS
44 =HRS ;Q_#MONTH =MONTH ; IF ( MONTH = 'JAN') || ( MONTH = '
45 MAR') || ( MONTH = 'MAY') || ( MONTH = 'JUL') || ( MONTH = 'AUG') || ( MONTH = 'OCT') || ( MONTH = 'DEC') THEN DO; DAYLIM = 31; GO TO Q1#001; END; IF ( MONTH = 'APR') || ( MONTH = 'JUN') || ( MONTH = 'SEP') || ( MONTH = 'NOV') THEN DO; DAYLIM = 30; GO TO Q1#001; END; IF ( MONTH = 'FEB') THEN DO; DAYLIM = 29; GO TO Q1#001; END; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='MONTH INVALID'; GO TO TST2; Q1#001: IF ( DAY = 0 ) || ( DAY > DAYLIM ) THEN DO; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='DAY INVALID'; GO TO Q1#002; END ; TST2: Q1#002: IF ( MIN > MAX ) THEN DO; IF ( MAX = 0 ) THEN DO; GO TO Q2#001; END; Q_#FATAL=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='MIN > MAX'; GO TO Q1#003; END; Q1#003: Q2#001: IF ( MAX = 0 ) THEN DO; IF MONTH= 'JAN' THEN DO; MAX = 82.4; GO TO Q_#L001; END; IF MONTH= 'FEB' THEN DO; MAX = 80.7; GO TO Q_#L001; END; IF MONTH= 'MAR' THEN DO; MAX = 76.2; GO TO Q_#L001; END ; IF MONTH= 'APR' THEN DO; MAX = 66.7; GO TO Q_#L001; END; IF MONTH= 'MAY' THEN
```



```
DO; MAX = 59.3; GO TO Q #L001; END; IF MONTH= 'JUN' THEN DO; MAX = 52.6; GO TO Q
#L001; END; IF MONTH= 'JUL' THEN DO; MAX = 51.8; GO TO Q #L001; END; IF MONTH=
'AUG' THEN DO; MAX = 55.1; GO TO Q #L001; END; IF MONTH= 'SEP' THEN DO; MAX = 61
.4; GO TO Q #L001; END; IF MONTH= 'OCT' THEN DO; MAX = 67.0; GO TO Q #L001; END;
IF MONTH= 'NOV' THEN DO; MAX = 72.9; GO TO Q #L001; END; IF MONTH= 'DEC' THEN D
O; MAX = 79.2; GO TO Q #L001; END; MAX = 0; Q #L001: Q #ADJUST=Q #ADJUST+1; IF Q
#DIGCT=40 THEN Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='MAX TEMP NOT STATED'; GO
TO Q1#004; END; IF MONTH= 'JAN' THEN DO; MAXLIM = 110.0; GO TO Q #L002; END; I
F MONTH= 'FEB' THEN DO; MAXLIM = 100.0; GO TO Q #L002; END; IF MONTH= 'MAR' THEN
DO; MAXLIM = 100.0; GO TO Q #L002; END; IF MONTH= 'APR' THEN DO; MAXLIM = 90.0;
GO TO Q #L002; END; IF MONTH= 'MAY' THEN DO; MAXLIM = 75.0; GO TO Q #L002; END;
IF MONTH= 'JUN' THEN DO; MAXLIM = 65.0; GO TO Q #L002; END; IF MONTH= 'JUL' THE
N DO; MAXLIM = 65.0; GO TO Q #L002; END; IF MONTH= 'AUG' THEN DO; MAXLIM = 72.0;
GO TO Q #L002; END; IF MONTH= 'SEP' THEN DO; MAXLIM = 82.0; GO TO Q #L002; END;
IF MONTH= 'OCT' THEN DO; MAXLIM = 90.0; GO TO Q #L002; END; IF MONTH= 'NOV' THE
N DO; MAXLIM = 102.0; GO TO Q #L002; END; IF MONTH= 'DEC' THEN DO; MAXLIM = 105.
0; GO TO Q #L002; END; MAXLIM = 100.0; Q #L002: IF ( MAX > MAXLIM + 5 ) THEN DO; Q
#FATAL=1; IF Q #DIGCT=40 THEN Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='MAXIMUM
TOO HIGH'; GO TO Q2#002; END; IF ( MAX > MAXLIM ) THEN DO; Q #QUERY=1; IF Q #DIGC
T=40 THEN Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='MAXIMUM HIGH'; GO TO Q2#002;
END; Q1#004: Q2#002: IF ( MIN = 0 ) THEN DO; IF MONTH= 'JAN' THEN DO; MIN = 56
.0; GO TO Q #L003; END; IF MONTH= 'FEB' THEN DO; MIN = 56.1; GO TO Q #L003; END;
IF MONTH= 'MAR' THEN DO; MIN = 52.7; GO TO Q #L003; END; IF MONTH= 'APR' THEN D
O; MIN = 45.5; GO TO Q #L003; END; IF MONTH= 'MAY' THEN DO; MIN = 39.1; GO TO Q
#L003; END; IF MONTH= 'JUN' THEN DO; MIN = 35.7; GO TO Q #L003; END; IF MONTH= '
JUL' THEN DO; MIN = 33.8; GO TO Q #L003; END; IF MONTH= 'AUG' THEN DO; MIN = 35.
4; GO TO Q #L003; END; IF MONTH= 'SEP' THEN DO; MIN = 38.9; GO TO Q #L003; END;
IF MONTH= 'OCT' THEN DO; MIN = 44.2; GO TO Q #L003; END; IF MONTH= 'NOV' THEN DO
; MIN = 48.7; GO TO Q #L003; END; IF MONTH= 'DEC' THEN DO; MIN = 53.3; GO TO Q #
L003; END; MIN = 0; Q #L003: Q #ADJUST=Q #ADJUST+1; IF Q #DIGCT=40 THEN Q #DIGCT
=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='MIN TEMP NOT STATED'; GO TO Q1#005; END; IF MON
TH= 'JAN' THEN DO; MINLIM = 38.0; GO TO Q #L004; END; IF MONTH= 'FEB' THEN DO; M
INLIM = 35.0; GO TO Q #L004; END; IF MONTH= 'MAR' THEN DO; MINLIM = 35.0; GO TO
Q #L004; END; IF MONTH= 'APR' THEN DO; MINLIM = 29.0; GO TO Q #L004; END; IF MON
TH= 'MAY' THEN DO; MINLIM = 22.0; GO TO Q #L004; END; IF MONTH= 'JUN' THEN DO; M
INLIM = 18.0; GO TO Q #L004; END; IF MONTH= 'JUL' THEN DO; MINLIM = 20.0; GO TO
Q #L004; END; IF MONTH= 'AUG' THEN DO; MINLIM = 21.0; GO TO Q #L004; END; IF MON
TH= 'SEP' THEN DO; MINLIM = 25.0; GO TO Q #L004; END; IF MONTH= 'OCT' THEN DO; M
INLIM = 29.0; GO TO Q #L004; END; IF MONTH= 'NOV' THEN DO; MINLIM = 32.0; GO TO
Q #L004; END; IF MONTH= 'DEC' THEN DO; MINLIM = 36.0; GO TO Q #L004; END; MINLIM
= 25.0; Q #L004: IF ( MIN < MINLIM - 5 ) THEN DO; Q #FATAL=1; IF Q #DIGCT=40 THE
N Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='MINIMUM TOO LOW'; GO TO Q2#003; END;
IF ( MIN < MINLIM ) THEN DO; Q #QUERY=1; IF Q #DIGCT=40 THEN Q #DIGCT=Q #DIGCT+1;
Q #DIAG(Q #DIGCT)='MINIMUM LOW'; GO TO Q2#003; END; Q1#005: Q2#003: IF ( FAL
L > 15 ) THEN DO; Q #FATAL=1; IF Q #DIGCT=40 THEN Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q
#DIGCT)='FALL TOO HIGH'; GO TO Q1#006; END; IF ( FALL > 5 ) THEN DO; Q #QUERY=1;
IF Q #DIGCT=40 THEN Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='FALL HIGH'; GO TO
Q1#006; END; Q1#006: IF ( HRS > 20 ) THEN DO; Q #FATAL=1; IF Q #DIGCT=40 THEN
Q #DIGCT=Q #DIGCT+1; Q #DIAG(Q #DIGCT)='HOURS TOO HIGH'; GO TO Q1#007; END; Q1#
007: IF Q #ADJUST>1 THEN Q #FATAL=1; IF Q #FATAL THEN DO; FLAG =3; MAX
Q #MAX ;MIN =Q #MIN ;FALL =Q #FALL ;DAY
=Q #DAY ;HRS =Q #HRS ;MONTH =Q #MONTH
; PUT FILE (SYSPRINT) EDIT (' RECORD IN ERROR') (SKIP(2),A(16)); GO TO Q #COMPLT
; END; IF Q #QUERY THEN DO; FLAG =2; PUT FILE (SYSPRINT) EDIT (' RECORD QUERIED')
(SKIP(2),A(15)); GO TO Q #COMPLT; END; IF Q #ADJUST THEN DO; FLAG =1; PUT FILE (
SYSPRINT) EDIT (' RECORD ADJUSTED') (SKIP(2),A(16)); GO TO Q #COMPLT; END; FLAG=0
; Q #COMPLT: IF (Q #FATAL)|(Q #QUERY)|(Q #ADJUST) THEN DO; PUT FILE (SYSPRINT) EDI
```

WEATHER: PROCEDURE OPTIONS (MAIN);

PAGE 4

419
423
425

I (DAY,MONTH,MAX,MIN,FALL,HRS)(R(Q_#FORM)); END; IF Q_#DIGCT=0 THEN DO; PUT FILE
E (SYSPRINT); EDIT ((Q_#DIAG(IQ_#) DO IQ_#=1 TO Q_#DIGCT)) (SKIP,A(40)); END; END
EDITOR ; END WEATHER;

26441

FRED PRE-PROCESSOR ***

PAGE 4

END WEATHER;

END OF THIS COMPILATION, ERRORS WERE NOTED AS SIGNALLED ABOVE.
~~DO NOT ATTEMPT TO RUN THIS PROGRAM UNTIL THESE ERRORS HAVE BEEN REMOVED.~~

WEATHER: PROCEDURE OPTIONS (MAIN);

/* FIRST SAMPLE EDIT TO DEMONSTRATE FRED PRE-PROCESSOR
EDITS TEMPERATURE AND RAINFALL CARDS */

DECLARE (MAX,MAXTOT,MIN,MINTOT,FALL,FALLTOT,HRSTOT) FIXED DECIMAL(7,2)

INITIAL (0),

(DAY,HRS,TALLY) FIXED DECIMAL (3,0) INITIAL (0),

CARD CHARACTER(80),

MONTH CHARACTER (3), FLAG FIXED BINARY;

ON ENDFILE (SYSIN) GO TO COMPLETE;

PUT FILE (SYSPRINT) SKIP(60);

/* READ NEXT CARD AND SUBMIT TO EDIT */

READCARD: GET FILE (SYSIN) EDIT (CARD) (A(80));

GET STRING (CARD) EDIT (DAY,MONTH,MAX,MIN,FALL,HRS)

(F(2),A(3),3 F(5,2),F(2));

CALL EDITOR;

IF FLAG=3 THEN GO TO READCARD; /* DO NOT USE ERRORS */

TALLY=TALLY+1; MAXTOT=MAXTOT+MAX; MINTOT=MINTOT+MIN;

FALLTOT=FALLTOT+FALL; HRSTOT=HRSTOT+HRS;

GO TO READCARD;

/* ALL CARDS READ, COMPUTE AND PRINT AVERAGES */

COMPLETE: MAXTOT=MAXTOT/TALLY; MINTOT=MINTOT/TALLY;

HRSTOT=HRSTOT/TALLY;

PUT FILE (SYSPRINT) EDIT ('RESULTS FOR ',TALLY,' ACCEPTED',

' RECORDS ARE', 'AVERAGE MAXIMUM TEMP',MAXTOT,

'AVERAGE MINIMUM TEMP',MINTOT,'TOTAL RAINFALL',FALLTOT,

'AVERAGE HOURS SUNSHINE',HRSTOT)

(PAGE,X(2),A(12),F(4),A(9),A(12),SKIP,A(20),X(5),F(7,3),

SKIP,A(20),X(5),F(7,3),SKIP,A(14),X(11),F(7,3),SKIP,

A(22),X(3),F(7,3));

/* INCLUDE EDIT AS INTERNAL PROCEDURE */

START EDIT EDITOR

EDIT (MAX,MIN,FALL) FIXED DECIMAL (7,2),(DAY,HRS) FIXED DECIMAL

C(3,0), MONTH CHARACTER (3)

FLAG FLAG

LIST DAY,MONTH,MAX,MIN,FALL,HRS

FORMAT (SKIP,F(2),A(3),3 F(8,2),F(4))

MAXADJ 1

DECLARE (MAXLIM,MINLIM) FIXED DECIMAL (7,2),DAYLIM FIXED DECIMAL

C(3,0)

/* CHECK MONTH, AND GET NUMBER OF DAYS IN MONTH */

IF MONTH='JAN',='MAR',='MAY',='JUL',='AUG',='OCT',='DEC' THEN

DAYLIM=31

='APR',='JUN',='SEP',='NOV' THEN

DAYLIM=30

='FEB' THEN

DAYLIM=29

ELSE

ERROR 'MONTH INVALID'

GO TO TST2

IF DAY=0,>DAYLIM THEN /* TEST DAY */

ERROR 'DAY INVALID'

ELSE

**

FRED PRE-PROCESSOR ***

/* CHECK THEN MAXIMUM TEMP IS PRESENT AND IN RANGE */

TST2 IF MIN>MAX THEN

IF MAX=0 THEN /* OMIT MAX NOT STATED FROM THIS TEST */

**

ELSE

ERROR 'MIN > MAX'

ELSE

**

IF MAX=0 THEN /* NOT STATED, INSERT A VALUE */

LOOK UP MONTH TO MAX,

C = 'JAN', 82.4, = 'FEB', 80.7, = 'MAR', 76.2, = 'APR', 66.7,

C = 'MAY', 59.3, = 'JUN', 52.6, = 'JUL', 51.8, = 'AUG', 55.1,

C = 'SEP', 61.4, = 'OCT', 67.0, = 'NOV', 72.9, = 'DEC', 79.2,

C OTHER 0, 'MAX TEMP NOT STATED'

ELSE /* GET UPPER LIMIT FOR TEST */

LOOK UP MONTH FOR MAXLIM,

C = 'JAN', 110.0, = 'FEB', 100.0, = 'MAR', 100.0, = 'APR', 90.0,

C = 'MAY', 75.0, = 'JUN', 65.0, = 'JUL', 65.0, = 'AUG', 72.0,

C = 'SEP', 82.0, = 'OCT', 90.0, = 'NOV', 102.0, = 'DEC', 105.0,

C OTHER 100.0

IF MAX>MAXLIM+5 THEN

ERROR 'MAXIMUM TOO HIGH'

>MAXLIM THEN

QUERY 'MAXIMUM HIGH'

ELSE

**

/* CHECK THAT MINIMUM TEMP IS PRESENT AND IN RANGE */

IF MIN=0 THEN /* NOT STATED, INSERT A VALUE */

LOOK UP MONTH TO MIN,

C = 'JAN', 56.0, = 'FEB', 56.1, = 'MAR', 52.7, = 'APR', 45.5,

C = 'MAY', 39.1, = 'JUN', 35.7, = 'JUL', 33.8, = 'AUG', 35.4,

C = 'SEP', 38.9, = 'OCT', 44.2, = 'NOV', 48.7, = 'DEC', 53.3,

C OTHER 0, 'MIN TEMP NOT STATED'

ELSE /* GET LOWER LIMIT FOR TEST */

LOOK UP MONTH FOR MINLIM,

C = 'JAN', 38.0, = 'FEB', 35.0, = 'MAR', 35.0, = 'APR', 29.0,

C = 'MAY', 22.0, = 'JUN', 18.0, = 'JUL', 20.0, = 'AUG', 21.0,

C = 'SEP', 25.0, = 'OCT', 29.0, = 'NOV', 32.0, = 'DEC', 36.0,

C OTHER 25.0

IF MIN<MINLIM-5 THEN

ERROR 'MINIMUM TOO LOW'

<MINLIM THEN

QUERY 'MINIMUM LOW'

ELSE

**

/* CHECK RAINFALL */

IF FALL > 15 THEN

ERROR 'FALL TOO HIGH'

>5 THEN

QUERY 'FALL HIGH'

ELSE

**

/* CHECK HOURS OF SUNSHINE */

IF HRS>20 THEN

ERROR 'HOURS TOO HIGH'

ELSE

**

END EDIT EDITOR

```

1
2
3 XXST:  PROCEDURE;
4        /* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67
5           TO SIGNAL AN UN-RECOGNISEABLE STATEMENT */
6
7 DECLARE (ENDFLAG  FIXED BINARY(1),
8          ERROR     FIXED BINARY(1) ) STATIC EXTERNAL;
9
10        PUT FILE (SYSPRINT) EDIT (' *** STATEMENT TYPE CANNOT BE ',
11          'RECOGNISED') (SKIP,A(30),A(10));
12
13 LOOP:   CALL ENTITY; IF ENDFLAG=1 THEN GO TO LOOP;
14         RETURN;
15
16 END XXST;
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```


WEATHER: PROCEDURE OPTIONS (MAIN);

/* FIRST SAMPLE EDIT TO DEMONSTRATE FRED PRE-PROCESSOR
EDITS TEMPERATURE AND RAINFALL CARDS */DECLARE (MAX,MAXTOT,MIN,MINTOT,FALL,FALLTOT,HRSTOT) FIXED DECIMAL(7,2)
INITIAL (0),
(DAY,HRS,TALLY) FIXED DECIMAL (3,0) INITIAL (0),
CARD CHARACTER(80),
MONTH CHARACTER (3), FLAG FIXED BINARY;ON ENDFILE (SYSIN) GO TO COMPLETE;
PUT FILE (SYSPRINT) SKIP(60);/* READ NEXT CARD AND SUBMIT TO EDIT */
READCARD: GET FILE (SYSIN) EDIT (CARD) (A(80));
GET STRING (CARD) EDIT (DAY,MONTH,MAX,MIN,FALL,HRS)
(F(2),A(3),3 F(5,2),F(2));
CALL EDITOR;
IF FLAG=3 THEN GO TO READCARD; /* DO NOT USE ERRORS */
TALLY=TALLY+1; MAXTOT=MAXTOT+MAX; MINTOT=MINTOT+MIN;
FALLTOT=FALLTOT+FALL; HRSTOT=HRSTOT+HRS;
GO TO READCARD;/* ALL CARDS READ, COMPUTE AND PRINT AVERAGES */
COMPLETE: MAXTOT=MAXTOT/TALLY; MINTOT=MINTOT/TALLY;
HRSTOT=HRSTOT/TALLY;
PUT FILE (SYSPRINT) EDIT ('RESULTS FOR ',TALLY,' ACCEPTED',
' RECORDS ARE',' AVERAGE MAXIMUM TEMP',MAXTOT,
' AVERAGE MINIMUM TEMP',MINTOT,' TOTAL RAINFALL',FALLTOT,
' AVERAGE HOURS SUNSHINE',HRSTOT)
(PAGE,X(2),A(12),F(4),A(9),A(12),SKIP,A(20),X(5),F(7,3),
SKIP,A(20),X(5),F(7,3),SKIP,A(14),X(11),F(7,3),SKIP,
A(22),X(3),F(7,3));

/* INCLUDE EDIT AS INTERNAL PROCEDURE */

START EDIT EDITOR

EDIT (MAX,MIN,FALL) FIXED DECIMAL (7,2), (DAY,HRS) FIXED DECIMAL
C(3,0), MONTH CHARACTER (3)

FLAG FLAG

LIST DAY,MONTH,MAX,MIN,FALL,HRS

FORMAT (SKIP,F(2),A(3),3 F(8,2),F(4))

MAXADJ 1

DECLARE (MAXLIM,MINLIM) FIXED DECIMAL (7,2), DAYLIM FIXED DECIMAL
C(3,0)

/* CHECK MONTH, AND GET NUMBER OF DAYS IN MONTH */

IF MONTH='JAN','MAR','MAY','JUL','AUG','OCT','DEC' THEN

DAYLIM=31

='APR','JUN','SEP','NOV' THEN

DAYLIM=30

='FEB' THEN

DAYLIM=29

IF DAY=0,>DAYLIM THEN /* TEST DAY */

*** NO ELSE STATEMENT TO IF=GROUP CLOSED BY THIS STATEMENT

ERROR 'DAY INVALID'

*** STARTING COLUMN IS ILLEGAL

ELSE

**

*** STARTING COLUMN IS ILLEGAL

FRED PRE-PROCESSOR ***

PAGE 2

/* CJECK THEN MAXIMUM TEMP IS PRESENT AND IN RANGE */

TST2 IF MIN>MAX THEN

IF MAX=0 THEN /* OMIT MAX NOT STATED FROM THIS TEST */

**

ELSE

ERROR 'MIN > MAX'

ELSE

**

IF MAX=0 THEN /* NOT STATED, INSERT A VALUE */

LOOK UP MONTH TO MAX,

C = 'JAN',82.4, = 'FEB',80.7, = 'MAR',76.2, = 'APR',66.7,

C = 'MAY',59.3, = 'JUN',52.6, = 'JUL',51.8, = 'AUG',55.1,

C = 'SEP',61.4, = 'OCT',67.0, = 'NOV',72.9, = 'DEC',79.2,

C OTHER 0, 'MAX TEMP NOT STATED'

ELSE /* GET UPPER LIMIT FOR TEST */

LOOK UP MONTH FOR MAXLIM,

C = 'JAN',110.0, = 'FEB',100.0, = 'MAR',100.0, = 'APR',90.0,

C = 'MAY',75.0, = 'JUN',65.0, = 'JUL',65.0, = 'AUG',72.0,

C = 'SEP',82.0, = 'OCT',90.0, = 'NOV',102.0, = 'DEC',105.0,

C OTHER 100.0

IF MAX>MAXLIM+5 THEN

ERROR 'MAXIMUM TOO HIGH'

>MAXLIM THEN

QUERY 'MAXIMUM HIGH'

ELSE

**

/* CHECK THAT MINIMUM TEMP IS PRESENT AND IN RANGE */

IF MIN=0 THEN /* NOT STATED, INSERT A VALUE */

LOOK UP MONTH TO MIN,

C = 'JAN',56.0, = 'FEB',56.1, = 'MAR',52.7, = 'APR',45.5,

C = 'MAY',39.1, = 'JUN',35.7, = 'JUL',33.8, = 'AUG',35.4,

C = 'SEP',38.9, = 'OCT',44.2, = 'NOV',48.7, = 'DEC',53.3,

C OTHER 0, 'MIN TEMP NOT STATED'

ELSE /* GET LOWER LIMIT FOR TEST */

LOOK UP MONTH FOR MINLIM,

C = 'JAN',38.0, = 'FEB',35.0, = 'MAR',35.0, = 'APR',29.0,

C = 'MAY',22.0, = 'JUN',18.0, = 'JUL',20.0, = 'AUG',21.0,

C = 'SEP',25.0, = 'OCT',29.0, = 'NOV',32.0, = 'DEC',36.0,

*** LOOK-UP STATEMENT INCOMPLETE

IF MIN<MINLIM-5 THEN

ERROR 'MINIMUM TOO LOW'

<MINLIM THEN

QUERY 'MINIMUM LOW'

ELSE

**

/* CHECK RAINFALL */

IF FALL > 15 THEN

ERROR 'FALL TOO HIGH'

>5 THEN

QUERY 'FALL HIGH'

ELSE

**

END EDIT EDITOR

94937

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

FRED PRE-PROCESSOR ***
LOGICAL CROSS-REFERENCE TABLE FOR THIS EDIT IS
MAX ***** MAX MIN MONTH
MIN ***** MAX MIN MONTH
FALL ***** FALL
DAY ***** DAY
HRS *****
MONTH ***** MAX MIN MONTH

50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

94936

```

1
2
3 STARTER: PROCEDURE;
4     /* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
5      TO READ AND COPY A PL/I PROGRAM AND SEARCH FOR A START
6      EDIT STATEMENT, THEN INITIATE FRED PROCEDURE */
7
8 DECLARE (ENDFLAG FIXED BINARY(1),
9         ENDPROG  FIXED BINARY(1),
10        EDITNAM  CHARACTER(7) ) STATIC EXTERNAL,
11        NAMHOLD(7) PACKED CHARACTER(1) DEFINED EDITNAM,
12        CARD(80) PACKED CHARACTER(1),
13        KEY      CHARACTER(16) DEFINED CARD,
14        ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA,
15        ALPHA    CHARACTER(29) STATIC INITIAL
16                ('ABCDEFGHIJKLMNOPQRSTUVWXYZ#@$');
17
18     /* READ CARDS UNTIL START EDIT STATEMENT */
19     ON ENDFILE (SYSIN) BEGIN;
20     GO TO ENDOFIT; END;
21
22 LOOP:  GET FILE (SYSIN) EDIT (CARD) (80 A(1));
23        IF KEY=' ' START EDIT THEN GO TO FOUND;
24        PUT FILE (OBJECT) EDIT ((CARD(I) DO I=2 TO 72)) (71 A(1));
25        PUT FILE (SYSPRINT) EDIT (CARD) (SKIP,80 A(1));
26        GO TO LOOP;
27
28 FOUND: /* EXTRACT EDIT NAME */
29        PUT FILE (SYSPRINT) EDIT (CARD) (SKIP, 80 A(1));
30        DO I=17 TO 80; IF CARD(I)=' ' THEN GO TO NAM; END;
31        PUT FILE (SYSPRINT) EDIT (' *** NO NAME HAS BEEN GIVEN TO ',
32        'THIS EDIT') (SKIP,A(31),A(9)); ERROR=1; RETURN;
33
34 NAM:    /* NAME MUST HAVE ALPHA FIRST CHARACTER, NOT EXCEED 7 CHARS */
35        DO J=1 TO 29; IF CARD(I)=ALPHABET(J) THEN GO TO OK; END;
36        PUT FILE (SYSPRINT) EDIT(' *** EDIT NAME DOES NOT HAVE ',
37        'ALPHABETIC FIRST CHARACTER') (SKIP,A(29),A(26));
38        ERROR=1; RETURN;
39
40 OK:     DO J=1 TO 7;
41         NAMHOLD(J)=CARD(I+J-1);
42         IF (CARD(I+J)=' ')|(CARD(I+J)='(') THEN GO TO ENDNAM;
43         END;
44         PUT FILE (SYSPRINT) EDIT (' *** EDIT NAME IS TOO LONG')
45         (SKIP,A(26)); ERROR=1; RETURN;
46
47 ENDNAM: K=J+1; DO L=K TO 7; NAMHOLD(L)=' '; END; /* PAD WITH BLANKS */
48        I=I+J; /* RESET I FOR PUT STATEMENT */
49
50     /* WRITE PROCEDURE STATEMENT INCLUDING PARAMETERS */
51     PUT FILE (OBJECT) EDIT (EDITNAM,' :PROCEDURE ',(CARD(J) DO
52     J=1 TO 72),';') (A(7),A(11),70 A(1));
53
54     /* WRITE DECLARE FOR PRIVATE FLAGS */
55     PUT FILE (OBJECT) EDIT
56     ('DECLARE Q_#DIAG(40) PACKED CHARACTER(32),',
57     ' (Q_#FATAL,Q_#QUERY) FIXED BINARY(1),',
58     ' (Q_#ADJUST,Q_#DIGCT) FIXED BINARY;') (A(41),A(36),A(34));
59     ENDPROG=0; RETURN;
60
61 ENDOFIT: ENDPROG=1; RETURN;
62
63 END STARTER;

```



```

1
2
3      PRST:  PROCEDURE;
4              /* FRED PRE-PROCESSOR, DAVID MNEDUS, 24/3/67,
5                 TO PROCESS A PRINTALL STATEMENT */
6
7      DECLARE (ENDFLAG  FIXED BINARY(1),
8                PRALFLG  FIXED BINARY(1)          ) STATIC EXTERNAL;
9
10             IF PRALFLG THEN DO; /* ONLY ONE PRINTALL STATEMENT */
11                 PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST ',
12                    'PRINTALL STATEMENT') (SKIP,A(27),A(18)); ERROR=1;
13                 GO TO BADEND;
14             END;
15             PRALFLG=1;
16             CALL ENTITY; IF ENDFLAG THEN RETURN; /* NO MORE LECAL */
17             PUT FILE ( SYSPRINT) EDIT (' *** PRINTALL STATEMENT FORMAT ',
18                'INVALID') (SKIP,A(31),A(7)); ERROR=1;
19             CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
20
21 BADEND:
22 END PRST;
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

```

1
2
3 STST: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 24/3/67
5 TO EDIT AND PROCESS A START SUBEDIT STATEMENT */
6
7 DECLARE (ENDFLAG FIXED BINARY(1),
8 SYNTAX(32) PACKED CHARACTER(1),
9 SYNCNT FIXED BINARY,
10 ERROR FIXED BINARY(1),
11 SUBFLG FIXED BINARY(1),
12 SUBNAME(32) PACKED CHARACTER(1),
13 SUBCNT FIXED BINARY,
14 COLONE FIXED BINARY ) STATIC EXTERNAL,
15 ALPHA CHARACTER(29) STATIC INITIAL
16 ('ABCDEFGHIJKLMNOPQRSTUVWXYZ#@$'),
17 ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA,
18 WORD CHARACTER(32) DEFINED SYNTAX;
19
20 IF COLONE=7 THEN DO; /* MUST BE IN FIRST COL OF MATRIX */
21 PUT FILE (SYSPRINT) EDIT (' *** START STATEMENT MUST BEGIN',
22 ' IN COLUMN 7') (SKIP,A(31),A(12)); ERROR=1; GO TO BADEND;
23 END;
24 CALL ENTITY; /* NEXT STATEMENT MUST BE SUBEDIT */
25 IF ENDFLAG THEN GO TO TOOSOON;
26 IF WORD='SUBEDIT' THEN GO TO BADFORM;
27 IF SUBFLG THEN DO; /* NESTED SUBEDITS ARE ILLEGAL */
28 PUT FILE (SYSPRINT) EDIT (' *** THIS STATEMENT OCCURS IN ',
29 'A SUBEDIT, DELETED') (SKIP,A(30),A(18)); ERROR=1;
30 GO TO BADEND; END;
31 CALL ENTITY; /* NEXT UNIT IS SUBEDIT NAME */
32 IF ENDFLAG THEN GO TO TOOSOON;
33 IF SYNCNT>7 THEN GO TO BADNAME; /* CHECK VALID NAME FORMAT */
34 DO I=1 TO 29; IF SYNTAX(I)=ALPHABET(I) THEN GO TO OK; END;
35 BADNAME: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF SUBEDIT NAME IS ',
36 'INVALID') (SKIP,A(31),A(7)); ERROR=1; GO TO BADEND;
37
38 OK: SUBFLG=1; SUBNAME=SYNTAX; SUBCNT=SYNCNT; /* STORE NAME */
39 PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
40 (X(1),8A(1)); PUT FILE (OBJECT) EDIT (': PROCEDURE;')
41 (A(12));
42 CALL ENTITY; /* SHOULD BE NO MORE */
43 IF ENDFLAG THEN RETURN;
44 BADFORM: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF START SUBEDIT ',
45 'STATEMENT IS INVALID') (SKIP,A(29),A(20)); ERROR=1;
46 BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
47 TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** START SUBEDIT STATEMENT ',
48 'IS INCOMPLETE') (SKIP,A(29),A(13));
49
50 END STST;

```



```

DO I=1 TO 6; IF WORD=SPECIAL(I) THEN GO TO OK3; END;
GO TO BADFORM; /* NOT LOGICAL OPERATOR */

/* READ FIRST OF PAIR AND WRITE CODE */
OK3: PUT FILE (OBJECT) EDIT (' IF ',(HOLD1(I) DO I=1 TO CNT1),
(SYNTAX(I) DO I=1 TO SYNCNT)) (A(4),34 A(1));
LOOP1: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(1)=',' THEN GO TO SECOND; /* END OF FIRST EXP */
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1)); IF SYNTAX(1)~='(' THEN GO TO LOOP1;
CALL BRACKET; GO TO LOOP1;

/* PROCESS SECOND EXPRESSION OF PAIR */
SECOND: IF TYPEFLG=-1 THEN GO TO GETLAB; /* IF NOT LABEL FORM ASS. */
PUT FILE (OBJECT) EDIT (' THEN DO;') (A(9));
PUT FILE (OBJECT) EDIT ((HOLD2(I) DO I=1 TO CNT2))
(X(1),32 A(1));
LOOP2: PUT FILE (OBJECT) EDIT (' =') (A(2)); /* NOW COPY REST */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(1)=',' THEN GO TO FINTHIS;
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1));
IF SYNTAX(1)~='(' THEN CALL BRACKET;
GO TO LOOP2;
FINTHIS: PUT FILE (OBJECT) EDIT ('; GO TO ',UPLAB,'; END;')
(A(8),A(7),A(6)); GO TO MAINLOOP;
/* JUMP, READ LABEL AND WRITE JUMP */
GETLAB: PUT FILE (OBJECT) EDIT (' THEN GO TO ') (A(12));
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
DO I=1 TO 29; IF SYNTAX(1)=ALPHABET(I) THEN GO TO OKLAB; END;
BADLAB: PUT FILE (SYSPRINT) EDIT (' *** LABEL GIVEN IN LIST HAS ',
'INVALID FORMAT') (SKIP,A(29),A(14)); ERROR=1; GO TO BADEND;
OKLAB: PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT),';')
(33 A(1));
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(1)=',' THEN GO TO MAINLOOP; ELSE GO TO BADLAB;

/* OTHER DETECTED, END IS HIGH */
OTHERF: IF TYPEFLG=-1 THEN GO TO LASTLAB;
PUT FILE (OBJECT) EDIT ((HOLD2(I) DO I=1 TO CNT2))
(X(1),32 A(1));
PUT FILE (OBJECT) EDIT (' =') (A(2));
LASTLOOP: CALL ENTITY; IF ENDFLAG THEN GO TO ONEGOOD;
IF SYNTAX(1)=',' THEN GO TO NILGOOD;
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1));
IF SYNTAX(1)~='(' THEN CALL BRACKET;
GO TO LASTLOOP;
ONEGOOD: IF TYPEFLG=-1 THEN GO TO TOOSOON; GO TO WRITLAST;
NILGOOD: IF TYPEFLG=0 THEN GO TO BADFORM;
WRITLAST: PUT FILE (OBJECT) EDIT ('; ',UPLAB,';') (A(1),A(7),A(1));
UPCNT=UPCNT+1; /* UPDATE LABEL FOR THIS STATEMENT */
PUT STRING (UPLAB) EDIT ('Q_#L',UPCNT) (A(4),F(3,0));
IF TYPEFLG=0 THEN DO;
PUT FILE (OBJECT) EDIT (' Q_#ADJUST=Q_#ADJUST+1;') (A(23));
CALL ADCON; END;
RETURN;

/* PROCESS END OF JUMP STATEMENT */
LASTLAB: PUT FILE (OBJECT) EDIT (' GO TO ') (A(6));

```

```

1
2
3      CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
4      DO I=1 TO 29; IF SYNTAX(I)=ALPHABET(I) THEN GO TO OKZ; END;
5      GO TO BADLAB;
6      OKZ:   PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT),';')
7            (X(1),33 A(1));
8            CALL ENTITY; IF ENDFLAG THEN RETURN;
9
10     /* GENERAL ERROR ROUTINE */
11     BADFORM: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF LOOK-UP INVALID')
12            (SKIP,A(30)); ERROR=1;
13     BADEND:  CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
14     TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** LOOK-UP STATEMENT INCOMPLETE')
15            (SKIP,A(33)); ERROR=1; RETURN;
16
17     END LOST;
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```



```

1
2
3 MAST: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 26/3/67
5 TO PROCESS A MAXADJ STATEMENT */
6
7 DECLARE (ENDFLAG FIXED BINARY(1),
8 SYNTAX(32) PACKED CHARACTER(1),
9 SYNCNT FIXED BINARY,
10 ERROR FIXED BINARY(1),
11 MAXVAL FIXED BINARY ) STATIC EXTERNAL,
12 NUM CHARACTER(9) INITIAL ('123456789') STATIC,
13 NUMBERS(9) PACKED CHARACTER(1) DEFINED NUM;
14
15 /* ONLY UNIT SHOULD BE FIXED POINT CONSTANT */
16 CALL ENTITY;
17 IF ENDFLAG THEN DO;
18 PUT FILE (SYSPRINT) EDIT (' *** MAXADJ STATEMENT IS ',
19 'INCOMPLETE') (SKIP,A(25),A(10)); ERROR=1; RETURN; END;
20 IF SYNCNT=1 THEN GO TO BADEND;
21 DO I=1 TO 9; IF SYNTAX(1)=NUMBERS(I) THEN GO TO OK; END;
22 GO TO BADEND;
23 OK: GET STRING (SYNTAX(1)) EDIT (MAXVAL) (F(1,0));
24 CALL ENTITY; IF ENDFLAG THEN RETURN;
25 BADEND: PUT FILE (SYSPRINT) EDIT (' *** NUMBER IS NOT INTEGER 1-9')
26 (SKIP,A(30)); ERROR=1;
27 BAD: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BAD; RETURN;
28 END MAST;
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

NEXTONE: PROCEDURE;

/* FRED PRE-PROCESSOR, DAVID MENDUS, 19/2/67
PART OF THE RECOGNISER TO RETURN THE NEXT SIGNIFICANT
CHARACTER. PROCESSES CONTINUATION CARDS AND READS OVER
COMMENTS. */

DECLARE (COLCNT FIXED BINARY,
COLONE FIXED BINARY,
ENDFLAG FIXED BINARY(1),
INWORD FIXED BINARY(1),
REPEAT FIXED BINARY(1),
ERROR FIXED BINARY(1),
INTEND FIXED BINARY(1) INITIAL(0),
CHARA CHARACTER(1))STATIC EXTERNAL,

(COMMENT FIXED BINARY(1) INITIAL(0),
APOSFLG FIXED BINARY(1) INITIAL(0),
CARD(80) PACKED CHARACTER(1),
ALPHABET CHARACTER(41) INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ\$@#._0123456789')
)STATIC,

SCANLIST(41) PACKED CHARACTER(1) DEFINED ALPHABET;

/*-----*/

IF INTEND THEN DO;

INTEND=0; ENDFLAG=1; RETURN; END;

IF REPEAT THEN DO;

/*IF REPEAT FLAG IS SET, DO - */

REPEAT=0; END;

/*NOT INCREMENT COLUMN COUNT. */

ELSE DO;

IF COLCNT=0 THEN DO;

/*IF NEW CARD, THEN PRINT IT */

PRINTIM: PUT FILE (SYSPRINT) EDIT ((CARD(I) DO I=1 TO 80))
(SKIP,X(1),80A(1));

COLONE=0; END;

INCRCT: COLCNT=COLCNT+1; END; /*INCREMENT COLUMN COUNT */

IF COLCNT<73 THEN GO TO NEXTCHAR;

/* NEW CARD NEEDED, READ AND CHECK FOR VALID CONTINUATION */
NEWCARD: GET FILE (SYSIN) EDIT ((CARD(I) DO I=1 TO 80)) (80A(1));

IF CARD(6)=' ' THEN DO;

/*NOT CONTINUATION CARD */

IF COLONE=0 THEN DO;

/*LAST CARD WAS NOT SIGNIFICANT, */

COLCNT=0; GO TO PRINTIM; /*DO NOT RETURN */

END;

ENDSTA: IF INWORD=0 THEN DO; /*DO NOT SIGNAL END OF CARD YET */

INWORD=0; INTEND=1; CHARA=' '; RETURN; END;

ENDFLAG=1;

/*SIGNAL END OF STATEMENT */

INWORD=0; COMMENT=0; COLCNT=0;

RETURN; END;

PUT FILE (SYSPRINT) EDIT ((CARD(I) DO I=1 TO 80))

(SKIP,X(1),80A(1));

/*PRINT CONTINUATION CARD */

DO COLCNT=1 TO 5,7 TO 72; /*FIND FIRST NON-BLANK COLUMN */

IF CARD(COLCNT)=' ' THEN GO TO FOUND;

END;

/*IF NONE COL 73 WILL RAISE ERROR*/

FOUND: IF COLCNT=COLONE THEN GO TO NEXTCHAR;

IF COLONE=0 THEN DO;

/*IF WE ARE IN CONTINUED COMMENT*/

IF COLCNT=73 THEN GO TO NEWCARD; /*THEN COLONE WON'T BE SET*/

ELSE GO TO NEXTCHAR;

/*DO NOT FLAG ERROR */

END;

/* LOOP IGNORING ALL SUCCEEDING CONTINUATION CARDS */
REJECT: PUT FILE (SYSPRINT) EDIT (' *** ILLEGAL CONTINUATION CARD, ',


```

1      'CONTENTS IGNORED') (SKIP,A(32),A(16));
2      ERROR=1;
3      GET FILE (SYSIN) EDIT ((CARD(I) DO I=1 TO 80)) (80A(1));
4      IF CARD(6)=' ' THEN GO TO ENDSTA; /*SIGNAL END OF STATEMENT */
5      PUT FILE (SYSPRINT) EDIT ((CARD(I) DO I=1 TO 80))
6      (SKIP,X(1),80A(1));
7      GO TO REJECT;
8
9      /* STORE AND PROCESS NEXT CHARACTER */
10     NEXTCHAR:CHARA=CARD(COLCNT); /*STORE CHARACTER FOR RETURN */
11     IF APOSFLG THEN DO;
12     IF CHARA=''' THEN APOSFLG=0;
13     INWORD=1; RETURN; END;
14     IF CHARA=' ' THEN DO; /*PROCESS A BLANK CHARACTER */
15     IF INWORD=0 THEN GO TO INCRCT; /*IF EXCESS BLANK, IGNORE */
16     ELSE DO;
17     INWORD=0; RETURN; END; /*SIGNAL BLANK AND RETURN */
18     END;
19
20     /* CHECK FOR START OR FINISH OF A COMMENT */
21     IF COMMENT THEN DO; /* IN A COMMENT */
22     IF CHARA-='*' THEN GO TO INCRCT; /* NOT END, IGNORE */
23     IF COLCNT=72 THEN GO TO INCRCT; /*COMMENT CANNOT END -*/
24     COLCNT=COLCNT+1; /*OVER CARD BOUNDARY */
25     IF CARD(COLCNT)-='/' THEN GO TO NEXTCHAR;
26     COMMENT=0; GO TO INCRCT; /*END OF COMMENT FOUND*/
27     END;
28     IF CHARA='/' THEN DO; /*LOOK FOR THE START OF COMMENT */
29     IF COLCNT=72 THEN GO TO NOCOMM;
30     IF CARD(COLCNT+1)-='*' THEN GO TO NOCOMM;
31     COMMENT=1; /*START OF COMMENT */
32     COLCNT=COLCNT+1;
33     IF INWORD=0 THEN GO TO INCRCT; /*IF PRECEDED BY BLANKS DO-*/
34     INWORD=0;CHARA=' '; /*NOT SIGNAL, ELSE SIGNAL -*/
35     RETURN; /*BLANK (END OF WORD) */
36     END;
37
38     /*DECIDE TYPE OF CHARACTER AND SIGNAL ACCORDINGLY */
39     NOCOMM: DO I=1 TO 41;
40     IF CHARA=SCANLIST(I) THEN DO;
41     INWORD=1; RETURN; END; /*SIGNAL ALPHAMERIC */
42     END;
43     IF INWORD-1 THEN DO; /*SPECIAL CHARACTER */
44     IF CHARA=''' THEN DO;
45     APOSFLG=1; INWORD=1; RETURN; END;
46     INWORD=-1; RETURN; END; /*IF OUTSIDE WORD, SIGNAL */
47     INWORD=0; CHARA=' '; /*IF INSIDE WORD SIGNAL DUMMY -*/
48     REPEAT=1; RETURN; /*BLANK AND SET REPEAT FLAG */
49     END NEXTONE;

```

```

1
2
3      PEST:      PROCEDURE;
4                  /* FRED PRE-PROCESSOR, DAVID MENDUS, 24/3/67
5                     TO PROCESS A PERFORM STATEMENT */
6
7      DECLARE (ENDFLAG    FIXED BINARY(1),
8                SYNTAX(32) PACKED CHARACTER(1),
9                SYNCNT    FIXED BINARY(1),
10               ERROR      FIXED BINARY(1) ) STATIC EXTERNAL,
11               ALPHA      CHARACTER(29) STATIC INITIAL
12                      ('ABCDEFGHIJKLMNOPQRSTUVWXYZ#@$',),
13               ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA;
14
15      CALL ENTITY; /* NEXT UNIT SHOULD BE SUBEDIT NAME */
16      IF ENDFLAG THEN DO;
17          PUT FILE (SYSPRINT) EDIT (' *** NO SUBEDIT IS SPECIFIED')
18              (SKIP,A(28)); ERROR=1; RETURN; END;
19      IF SYNCNT<8 THEN DO; /* CHECK FORMAT OF SUBEDIT NAME */
20          DO I=1 TO 29; IF SYNTAX(1)=ALPHABET(I) THEN GO TO OK; END;
21      END; /* NAME FORMAT BAD */
22      PUT FILE (SYSPRINT) EDIT (' *** SUBEDIT NAME FORMAT INVALID')
23          (SKIP,A(32)); ERROR=1; GO TO BADEND;
24
25      OK:      PUT FILE (OBJECT) EDIT (' CALL ',(SYNTAX(I) DO I=1 TO SYNCNT),
26              ';;') (A(6),8 A(1));
27      CALL ENTITY; IF ENDFLAG THEN RETURN; /* SHOULDE END HERE */
28      PUT FILE (SYSPRINT) EDIT (' *** PERFORM STATEMENT FORMAT ',
29              'INVALID') (SKIP,A(30),A(7)); ERROR=1;
30
31      BADEND:  CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
32      END PEST;
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```



```

1
2
3      PNST:  PROCEDURE;
4              /* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67
5              TO PROCESS A PRINTNONE STATEMENT */
6
7      DECLARE (ENDFLAG  FIXED BINARY(1),
8              ERROR     FIXED BINARY(1),
9              LISTFLG   FIXED BINARY(1)) ) STATIC EXTERNAL;
10
11             /* THERE MAY BE ONLY ONE LIST OR PRINTNONE STATEMENT */
12             IF LISTFLG=0 THEN DO;
13                 PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST ',
14                 'LIST OR PRINTNONE STATEMENT') (SKIP,A(27),A(27));
15                 ERROR=1; END;
16             ELSE LISTFLG=-1;
17             CALL ENTITY; IF ENDFLAG=1 THEN RETURN;
18             PUT FILE (SYSPRINT) EDIT (' *** PRINTNONE FORMAT ILLEGAL')
19             (SKIP,A(29)); ERROR=1;
20             LOOP:  CALL ENTITY; IF ENDFLAG=0 THEN GO TO LOOP;
21             RETURN;
22     END PNST;
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

1
2
3 LAST: PROCEDURE:
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 16/3/67
5 TO PROCESS A LABEL TO A STATEMENT */
6 DECLARE (SYNTAX(32) PACKED CHARACTER(1),
7 SYNCNT FIXED BINARY)STATIC EXTERNAL;
8 PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT),':')
9 (X(1),6 A(1));
10 END LAST;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50


```

1
2
3 LIST: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
5 TO PROCESS A LIST STATEMENT */
6
7 DECLARE (ENDFLAG FIXED BINARY(1),
8 SYNTAX(32) PACKED CHARACTER(1),
9 SYNCNT FIXED BINARY,
10 ERROR FIXED BINARY(1),
11 LISTFLG FIXED BINARY(1),
12 LISTCNT FIXED BINARY ) STATIC EXTERNAL,
13
14 LISTSTR(LISTCNT) PACKED CHARACTER(1) CONTROLLED EXTERNAL,
15 LISTTEMP(2000) PACKED CHARACTER(1);
16 IF LISTFLG THEN DO; /* ONLY ONE LIST OR PRINTNONE ALLOWED */
17 PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST ',
18 'LIST OR PRINTNONE STATEMENT') (SKIP,A(27),A(27));
19 ERROR=1; GO TO BADEND; END;
20
21 LOOP: LISTFLG=1; LISTCNT=0; /* INITIALISE COUNT */
22 CALL ENTITY; IF ENDFLAG THEN DO; /* MOVE LIST TO EXT.STORE */
23 ALLOCATE LISTSTR; DO I=1 TO LISTCNT;
24 LISTSTR(I)=LISTTEMP(I); END; RETURN;
25 END;
26
27 IF LISTCNT+SYNCNT>2000 THEN DO; /* NO ROOM TO STORE UNIT */
28 PUT FILE (SYSPRINT) EDIT (' *** LENGTH OF LIST STATEMENT ',
29 'EXCEEDS COMPILER LIMIT') (SKIP,A(30),A(22)); ERROR=1;
30
31 BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
32 END;
33 DO I=1 TO SYNCNT;
34 LISTTEMP(LISTCNT+I)=SYNTAX(I);
35 END;
36 LISTCNT=LISTCNT+SYNCNT; GO TO LOOP;
37
38 END LIST;
39
40
41
42
43
44
45
46
47
48
49
50

```

```

LOST: PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS, 24/3/67
TO PROCESS LOOK UP STATEMENTS OF THREE KINDS. LOOK UP JUMP,
LOOK UP TO, LOOK UP BY. CALLS ADCON TO COMPLETE A ... TO
STATEMENT WITH DIAGNOSTIC ACTION. */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCNT FIXED BINARY,
ERROR FIXED BINARY(1),
NITEM FIXED BINARY ) STATIC EXTERNAL,
RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL,

(UPLAB CHARACTER(7) INITIAL('Q #LOO1'),
UPCNT FIXED BINARY INITIAL(1001),
TYPEFLG FIXED BINARY(1),
HOLD1(32) PACKED CHARACTER(1),
HOLD2(32) PACKED CHARACTER(1),
CNT1 FIXED BINARY,
CNT2 FIXED BINARY,
SP CHARACTER(12) INITIAL ('> < = >=<=<=')
) STATIC,
ALPHA CHARACTER(29) STATIC INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ#@$'),
SPECIAL(6) PACKED CHARACTER(2) DEFINED SP,
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA,
WORD CHARACTER(32) DEFINED SYNTAX;

/* STORE FIRST NAME AND DECIDE TYPE OF LOOKUP STATEMENT */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF (SYNCNT=2) | (SYNTAX(1)= 'U') | (SYNTAX(2)= 'P') THEN
GO TO BADFORM; /* NEXT WORD MUST BE UP */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
DO I=1 TO 29; IF SYNTAX(1)=ALPHABET(I) THEN GO TO OK1; END;
PUT FILE (SYSPRINT) EDIT (' *** FIRST VARIABLE NAME FORM AT ',
'ILLEGAL') (SKIP,A(32),A(7)); ERROR=1; GO TO BADEND;
OK1: HOLD1=SYNTAX; CNT1=SYNCNT; /* STORE FIRST NAME */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF (SYNTAX(1)= 'J') & (SYNTAX(2)= 'U') & (SYNTAX(3)= 'M')
& (SYNTAX(4)= 'P') THEN DO; /* TYPE IS JUMP, NO SECOND NAME */
TYPEFLG=-1; GO TO LOOKCOM; END;
IF (SYNTAX(1)= 'T') & (SYNTAX(2)= 'O') & (SYNCNT=2) THEN DO;
TYPEFLG=0; GO TO READTWO; END; /* TYPE IS TO, SECOND NAME*/
IF (SYNTAX(1)= 'F') & (SYNTAX(2)= 'O') & (SYNTAX(3)= 'R') THEN DO;
TYPEFLG=1; GO TO READTWO; END; /* TYPE IS BY, SECOND NAME*/
GO TO BADFORM; /* TYPE IS ILLEGAL */

/* READ AND STORE SECOND NAME, FIND COMMA */
READTWO: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF TYPEFLG=1 THEN GO TO OK2;
DO I=1 TO NITEM; IF WORD=RECORD(I) THEN GO TO OK2; END;
PUT FILE (SYSPRINT) EDIT (' *** SECOND VARIABLE IS NOT ',
'PART OF RECORD') (SKIP,A(28),A(14)); ERROR=1; GO TO BADEND;
OK2: HOLD2=SYNTAX; CNT2=SYNCNT; /* STORE SECOND NAME (NOT JUMP) */
LOOKCOM: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF WORD=',' THEN GO TO BADFORM;

/* START DECODING NEXT PAIR OF LOOKUP */
MAINLOOP: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF WORD='OTHER' THEN GO TO OTHER;

```

```

/* READ AND STORE SECOND NAME, FIND COMMA */
READTWO: CALL ENTITY; IF ENDFLAG THEN GO TO TOO SOON;
IF TYPEFLG=1 THEN GO TO OK2;
DO I=1 TO NITEM; IF WORD=RECORD(I) THEN GO TO OK2; END;
PUT FILE (SYSPRINT) EDIT (' *** SECOND VARIABLE IS NOT ',
'PART OF RECORD') (SKIP,A(28),A(14)); ERROR=1; GO TO BADEND;
OK2: HOLD2=SYNTAX; CNT2=SYNCNT; /* STORE SECOND NAME (NOT JUMP) */
LOOKCOM: CALL ENTITY; IF ENDFLAG THEN GO TO TOO SOON;
IF WORD=',' THEN GO TO BADFORM;

```

```

/* START DECODING NEXT PAIR OF LOOKUP */
MAINLOOP:CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF WORD='OTHER' THEN GO TO OTHER;

```



```

1      KEYWORD: PROCEDURE;
2      /* FRED PRE-PROCESSOR, DAVID MENDUS, 19/2/67
3      PART OF THE RECOGNISER TO BUILD AND IDENTIFY THE KEYWORD
4      FOR EACH FRED STATEMENT. */
5
6      DECLARE (COLCNT      FIXED BINARY,
7                COLONE     FIXED BINARY,
8                ENDFLAG    FIXED BINARY(1),
9                INWORD     FIXED BINARY(1),
10               REPEAT     FIXED BINARY(1),
11               ERROR      FIXED BINARY(1),
12               CHARA      CHARACTER(1),
13               SYNTAX(32) PACKED CHARACTER(1),
14               SYNCNT     FIXED BINARY,
15               FLAG       FIXED BINARY      )STATIC EXTERNAL,
16               WORD      CHARACTER(32) DEFINED SYNTAX,
17
18               RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL,
19               RECFLAG(NITEM) PACKED BIT(1) CONTROLLED EXTERNAL,
20               NITEM     FIXED BINARY STATIC EXTERNAL,
21
22               ALF       CHARACTER(29) STATIC INITIAL
23               ('ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@'),
24               ALPHABET(29) PACKED CHARACTER(1) DEFINED ALF;
25
26      /*-----*/
27      /* CLEAR FLAGS AND READ FIRST CHARACTER */
28      ENDFLAG=0;
29      RESTART: WORD=''; SYNCNT=0;
30      CALL NEXTONE;
31      IF INWORD=0 THEN GO TO RESTART;
32      IF COLCNT>5 THEN GO TO STMNT;
33
34      /* FIRST CHARACTER IS PART OF LABEL */
35      DO I=1 TO 29; /*TEST FOR ALPHABETIC FIRST CHARACTER */
36      IF CHARA=ALPHABET(I) THEN GO TO ALPHST;
37      END; /*ERROR MESSAGE IF NOT ALPHA FIRST CHA */
38      PUT FILE (SYSPRINT) EDIT (' *** LABEL FORMAT INVALID, ',
39      'LABEL HAS BEEN DELETED') (SKIP,A(27),A(22));
40      ERROR=1; /*FIX COUNTS TO IGNORE LABEL AND RESTART */
41      INWORD=0; COLCNT=6; GO TO RESTART;
42      ALPHST: SYNCNT=SYNCNT+1; SYNTAX(SYNCNT)=CHARA;
43      CALL NEXTONE;
44      IF INWORD THEN GO TO ALPHST; /*LOOP UNTIL LABEL ASSEMBLED */
45      FLAG=26; RETURN; /*SET LABEL FLAG AND RETURN */
46
47      STMNT: COLONE=COLCNT; /*SET STARTING COLUMN FLAG */
48      SYNCNT=1; SYNTAX(1)=CHARA;
49      IF INWORD=-1 THEN GO TO NOTSPEC;
50
51      /*SPECIAL CHARACTER, TEST FOR VALID COMBINATION */
52      IF (CHARA='=')|(CHARA='(') THEN DO; /* VALID - */
53      FLAG=28; RETURN; END; /*ONLY AS AN IF-STATEMENT: */
54      IF CHARA='*' THEN DO;
55      CALL NEXTONE;
56      IF CHARA='*' THEN DO;
57      SYNCNT=2; SYNTAX(2)=CHARA; FLAG=6; RETURN; END;
58      ELSE DO;
59      REPEAT=1; FLAG=29; RETURN; END;
60      END;
61      IF (CHARA='>')&(CHARA='<')&(CHARA='~') THEN DO;

```

```

1      FLAG=29; RETURN; END;
2      CALL NEXTONE;                /*LOOK FOR CHARACTER PAIR */
3      IF CHARA=' ' THEN DO;
4      SYNCNT=2; SYNTAX(2)=CHARA; /*VALID PAIR, SIGNAL AND RETURN */
5      FLAG=28; RETURN; END;
6      REPEAT=1;                    /*INVALID, SET FLAG SO CHARA IS NOT LOST */
7      IF SYNTAX(1)=' ' THEN DO;
8      FLAG=29; RETURN; END;        /*  INVALID UNLESS PART OF  = */
9      ELSE DO;
10     FLAG=28; RETURN; END;
11
12     /* FIRST CHARACTER MUST BE ALPHABETIC, ELSE ERROR */
13     NOTSPEC: DO I=1 TO 29;        /*TEST FOR ALPHABETIC FIRST CHARACTER */
14     IF CHARA=ALPHABET(I) THEN GO TO OKALPHA;
15     END;                          /*SIGNAL ERROR */
16     FLAG=29; RETURN;
17
18     OKALPHA: CALL NEXTONE;        /*VALID START, BUILD UP FIRST WORD */
19     IF INWORD=0 THEN GO TO ENDWORD;
20     IF SYNCNT=32 THEN DO;
21     FLAG=29; RETURN; END;        /*TOO LONG, SIGNAL ERROR */
22     SYNCNT=SYNCNT+1; SYNTAX(SYNCNT)=CHARA; GO TO OKALPHA;
23
24     /* FIRST WORD FORMED. COMPARE IT WITH ALL KEYWORDS */
25     ENDWORD: IF WORD='IF' THEN DO; FLAG= 1; RETURN; END;
26     IF WORD='ERROR' THEN DO; FLAG= 2; RETURN; END;
27     IF WORD='QUERY' THEN DO; FLAG= 3; RETURN; END;
28     IF WORD='ADJUST' THEN DO; FLAG= 4; RETURN; END;
29     IF WORD='ELSE' THEN DO; FLAG= 5; RETURN; END;
30     IF WORD='CONTINUE' THEN DO; FLAG= 6; RETURN; END;
31     IF WORD='**' THEN DO; FLAG= 6; RETURN; END;
32     IF WORD='LOOK' THEN DO; FLAG= 7; RETURN; END;
33     IF WORD='DO' THEN DO; FLAG= 8; RETURN; END;
34     IF WORD='GO' THEN DO; FLAG= 9; RETURN; END;
35     IF WORD='TRY' THEN DO; FLAG=10; RETURN; END;
36     IF WORD='DEFINE' THEN DO; FLAG=11; RETURN; END;
37     IF WORD='CHANGE' THEN DO; FLAG=12; RETURN; END;
38     IF WORD='PERFORM' THEN DO; FLAG=13; RETURN; END;
39
40     IF WORD='START' THEN DO; FLAG=14; RETURN; END;
41     IF WORD='END' THEN DO; FLAG=15; RETURN; END;
42     IF WORD='DECLARE' THEN DO; FLAG=16; RETURN; END;
43     IF WORD='EDIT' THEN DO; FLAG=17; RETURN; END;
44     IF WORD='FLAG' THEN DO; FLAG=18; RETURN; END;
45     IF WORD='MAXADJ' THEN DO; FLAG=19; RETURN; END;
46     IF WORD='LIST' THEN DO; FLAG=20; RETURN; END;
47     IF WORD='PRINTNONE' THEN DO; FLAG=21; RETURN; END;
48     IF WORD='FORMAT' THEN DO; FLAG=22; RETURN; END;
49     IF WORD='FILE' THEN DO; FLAG=23; RETURN; END;
50     IF WORD='HEADING' THEN DO; FLAG=24; RETURN; END;
51     IF WORD='PRINTALL' THEN DO; FLAG=25; RETURN; END;
52     /* 26=LABEL, 27=ASSIGNMENT OR IF, 28=IF, 29=ERROR */
53
54     /*NO MATCH, SIGNAL ASSIGNMENT AND UPDATE CROSS REF TABLE */
55     FLAG=27;
56     IF ALLOCATION(RECORD) THEN DO;
57     DO I=1 TO NITEM;
58     IF RECORD(I)=WORD THEN DO;
59     RECFLAG(I)='1'B; RETURN; END;
60     END;

```


END;
RETURN;
END KEYWORD;

94335

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
END;
NOTCOMP: LHSFLAG,STFLAG(MATRXCT)=0; /* END OF L.H.S. OPTION */
IF SYNTAX(1)='(' THEN DO; /* PARENTHESISED EXPRESSION */
  PUT FILE(OBJECT) EDIT ('(') (A(2));
  CALL BRACKET;
  IF ENDFLAG THEN GO TO TOSOON;
  GO TO STARTNXT;
END;
IF SYNTAX(1)=',' THEN GO TO BADFORM; /*, ILLEGAL HERE */

/* WRITE LHS IF NEEDED AND COMPARISON OPERATOR */
COMPEND: IF LHSFLAG THEN DO;
WRITE LHS: PUT FILE(OBJECT) EDIT ((LHSIDE(I) DO I=1 TO LHCNT))
  (X(1),32 A(1));
END;
PUT FILE(OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
  (X(1),32 A(1));

/* GET FIRST UNIT OF R.H.S. */
CALL ENTITY;
IF ENDFLAG THEN GO TO TOSOON;
IF SYNTAX(1)='(' THEN DO; /* PARENTHESISED EXPRESSION */
  PUT FILE(OBJECT) EDIT ('(') (A(2));
  CALL BRACKET; IF ENDFLAG THEN GO TO TOSOON;
  GO TO LATTER;
END;
IF (WORD=',')|(WORD='THEN') THEN GO TO BADFORM;
PUT FILE(OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
  (X(1),32 A(1));

/* READ ON TO END OF SUB-STATEMENT */
LATTER: CALL ENTITY; IF ENDFLAG THEN GO TO TOSOON;
KEEPON: IF SYNTAX(1)='(' THEN DO;
  PUT FILE(OBJECT) EDIT ('(') (A(2));
  CALL BRACKET; IF ENDFLAG THEN GO TO TOSOON;
  GO TO LATTER; END;
IF SYNTAX(1)=',' THEN DO; /*, ENDS SUB-STATEMENT */
  PUT FILE(OBJECT) EDIT (','|(') (A(3));
  GO TO STARTNXT; END;
IF WORD='THEN' THEN DO; /* THEN ENDS STATEMENT */
  CALL ENTITY; IF ENDFLAG=0 THEN DO; /* SPURIOUS - TREAT AS */
  PUT FILE(OBJECT) EDIT (' THEN') (A(5)); /* NAME */
  GO TO LATTER; END;
ELSE DO; /* GENUINE END OF STATEMENT */
  PUT FILE(OBJECT) EDIT (') THEN DO;') (A(10));
  RETURN; END;
END;
PUT FILE(OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
  (X(1),32 A(1));
GO TO LATTER;

/* ENTRY FOR SECONDARY IF-STATEMENTS */
I2ST: ENTRY; /* RESET L.H.S. CODES FOR THIS COLUMN */
LHWORD=STWORD(MATRXCT); LHCNT=STCNT(MATRXCT);
LHSFLAG=STFLAG(MATRXCT);
PUT FILE(OBJECT) EDIT (' END; IF(') (A(9));
GO TO EXTRAENT;

/* SET ERROR FLAG FOR INCOMPLETE STATEMENT */
TOSOON: PUT FILE(SYSPRINT) EDIT (' *** IF STATEMENT IS INCOMPLETE')

```

6239


```
1  
2  
3      (SKIP,A(31));  
4      ERROR=1; RETURN;  
5  
6      /* SET ERROR FLAG FOR BAD STRUCTURE */  
7      BADFORM: CALL ENTITY; IF ENDFLAG=1 THEN GO TO BADFORM;  
8      PUT FILE (SYSPRINT) EDIT (' *** IF-STATEMENT STRUCTURE IS ',  
9      'ILLEGAL') (SKIP,A(31),A(7));  
10     ERROR=1; RETURN;  
11     END IFST;  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50
```

END HEST;

176617


```

1
2
3 IFST: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 11/3/67
5 ROUTINE TO PROCESS ALL IF STATEMENTS, PRIMARY IFS FROM
6 ENTRY IFST, SECONDARY IFS FROM I2ST */
7
8 DECLARE (ENDFLAG FIXED BINARY(1),
9 SYNTAX(32) PACKED CHARACTER(1),
10 SYNCNT FIXED BINARY,
11 MATRXCT FIXED BINARY,
12 ERROR FIXED BINARY(1) ) STATIC EXTERNAL,
13 WORD CHARACTER(32) DEFINED SYNTAX,
14
15 (LHSFLAG FIXED BINARY(1),
16 LHSIDE(32) PACKED CHARACTER(1),
17 STFLAG(10) FIXED BINARY(1),
18 STWORD(10) PACKED CHARACTER(32),
19 STCNT(10) FIXED BINARY,
20 LHCNT FIXED BINARY ) STATIC,
21
22 SPWORD CHARACTER(12) INITIAL ('=><-=>=<=') STATIC,
23 SPCHA(6) PACKED CHARACTER(2) DEFINED SPWORD,
24 ALWORD CHARACTER(29) INITIAL
25 ('ABCDEFGHIJKLMNOPQRSTUVWXYZ#$%^') STATIC,
26 ALPHABET(29) PACKED CHARACTER(1) DEFINED ALWORD,
27 LHWORD CHARACTER(32) DEFINED LHSIDE;
28
29 /* START BY TESTING FIRST WORD FOR LEGALITY */
30 LHSFLAG,STFLAG(MATRXCT)=0;
31 PUT FILE (OBJECT) EDIT (' IF (') (A(5));
32 CALL ENTITY; /* GET NEXT UNIT */
33 IF ENDFLAG THEN GO TO TOSOON; /* STATEMENT CANT STOP HERE */
34 IF SYNTAX(1)='(' THEN DO; /* PARENTHESISED EXPRESSION */
35 PUT FILE (OBJECT) EDIT (' (') (A(2));
36 CALL BRACKET;
37 IF ENDFLAG THEN GO TO TOSOON;
38 ELSE GO TO STARTNXT;
39
40 END;
41 DO I=1 TO 29; /* IF NOT (-ISED THEN WORD MUST HAVE - */
42 IF SYNTAX(I)=ALPHABET(I) THEN GO TO STORE; /*ALPHA START*/
43 END;
44 GO TO BADFORM;
45
46 /* TEST FIRST WORD FOR A VALID L.H.S. */
47 STORE: LHSIDE=SYNTAX; STWORD(MATRXCT)=LHWORD;
48 LHCNT,STCNT(MATRXCT)=SYNCNT;
49 CALL ENTITY; /* NEXT UNIT MUST BE COMPARISON OPERATOR*/
50 IF ENDFLAG THEN GO TO TOSOON;
51 DO I=1 TO 6;
52 IF WORD=SPCHA(I) THEN DO;
53 LHSFLAG,STFLAG(MATRXCT)=1; GO TO WRITELHS; END;
54 END; /* CANNOT BE L.H.S., OUTPUT STORED UNIT*/
55 PUT FILE (OBJECT) EDIT ((LHSIDE(I) DO I=1 TO LHCNT)) (X(1),
56 32 A(1));
57 GO TO NOTCOMP;
58
59 /* READ L.H.S. OF NEXT SUB-STATEMENT IF PRESENT */
60 STARTNXT:CALL ENTITY;
61 IF ENDFLAG THEN GO TO TOSOON;
62 EXTRAENT:DO I=1 TO 6; /* TEST FOR COMPARISON OPERATOR */
63 IF WORD=SPCHA(I) THEN GO TO COMPEND;

```

GOST: PROCEDURE;
/* FRED PRE-PROCESSOR, DAVID MENDUS, 16/3/67
TO COMPILE A GO TO STATEMENT AND EDIT THE LABEL NAME */

DECLARE (ENDFLAG FIXED BINARY (1),
SYNTAX(32) PACKED CHARACTER (1),
SYNCNT FIXED BINARY,
ERROR FIXED BINARY (1)) STATIC EXTERNAL,
ALPHA CHARACTER (29) INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ#&') STATIC,
ALPHABET(29) PACKED CHARACTER (1) DEFINED ALPHA;

CALL ENTITY; /* NEXT UNIT MUST BE 'TO' */
IF ENDFLAG THEN GO TO TOOSOON;
IF (SYNCNT=2) || (SYNTAX(1)='T') || (SYNTAX(2)='O') THEN
GO TO BADSTR;

CALL ENTITY; /* THIRD UNIT MUST BE LABEL */
IF ENDFLAG THEN GO TO TOOSOON;
IF SYNCNT>5 THEN GO TO BADLAB; /* LABEL TOO LONG */
DO I=1 TO 29; /* FIRST CHARACTER MUST BE ALPHA */
IF SYNTAX(I)=ALPHABET(I) THEN GO TO WRITE; END;

BADLAB: PUT FILE (SYSPRINT) EDIT (' *** LABEL FORMAT INVALID')
(SKIP,A(25));

SKIPEND: CALL ENTITY; IF ENDFLAG THEN RETURN; GO TO SKIPEND;

WRITE: PUT FILE (OBJECT) EDIT (' GO TO',(SYNTAX(I) DO I=1 TO SYNCNT)
,',') (A(7),6 A(1));

CALL ENTITY; IF ENDFLAG THEN RETURN; /* ERROR IF MORE */
BADSTR: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF GO TO-STATEMENT',
' INVALID') (SKIP,A(30),A(8));
GO TO SKIPEND;

TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** GO TO-STATEMENT INCOMPLETE')
(SKIP,A(31));
RETURN;

END GOST;


```

1
2
3 HEST:  PROCEDURE;
4        /* FRED PRE-PROCESSOR, DAVID MENDUS, 26/3/67
5           TO EDIT AND PROSESS A HEADING STATEMENT, WRITING AN
6           ON ENDPAGE CONDITION ACTION, FILE MUST BE DECLARED */
7
8 DECLARE (ENDFLAG  FIXED BINARY(1),
9         CHARA     CHARACTER(1),
10        HEADFLG   FIXED BINARY(1),
11        FILEFLG   FIXED BINARY(1),
12        FILENAM(32) PACKED CHARACTER(1),
13        FILECNT   FIXED BINARY,
14        ERROR     FIXED BINARY(1),
15        INWORD    FIXED BINARY(1) ) STATIC EXTERNAL,
16        HEADING(133) PACKED CHARACTER (1),
17        HEADCNT   FIXED BINARY;
18
19        /* THERE MAY ONLY BE ONE HEADING DTATEMENT */
20        IF HEADFLG THEN DO;
21            PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST ',
22            'HEADING STATEMENT') (SKIP,A(27),A(17)); ERROR=1;
23            GO TO BADEND; END;
24        HEADFLG=1; /* FIRST SIGNIFICANT CHARACTER SHOULD BE ' */
25        CALL NEXTONE; IF ENDFLAG THEN DO;
26        TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** HEADING STATEMENT IS ',
27        'INCOMPLETE') (SKIP,A(26),A(10)); ERROR=1; RETURN;
28        END;
29        IF CHARA=''' THEN DO;
30        BADFORM: PUT FILE (SYSPRINT) EDIT (' *** HEADING STATEMENT FORMAT ',
31        'IS ILLEGAL') (SKIP,A(30),A(10)); ERROR=1; GO TO BADEND;
32        END;
33        HEADCNT=0; /* LOOP STORING CHARACTERS OF STRING */
34        CALL NEXTONE; IF ENDFLAG THEN GO TO TOOSOON;
35        IF CHARA=''' THEN GO TO ENDSTRING;
36        HEADCNT=HEADCNT+1;
37        IF HEADCNT>133 THEN DO;
38            PUT FILE (SYSPRINT) EDIT (' *** HEADING EXCEEDS 133 ',
39            'CHARACTERS') (SKIP,A(25),A(10)); ERROR=1; GO TO BADEND;
40        END;
41        HEADING(HEADCNT)=CHARA; GO TO LOOP;
42
43        /* END OF STRING, COULD BE ANOTHER STRING OR END */
44        ENDSTRING: CALL NEXTONE; IF ENDFLAG THEN GO TO ENDSTA;
45        IF INWORD=0 THEN GO TO ENDSTRING; /* INTERVENING BLANK */
46        IF CHARA=''' THEN GO TO LOOP; /* START NEXT STRING */
47        ELSE GO TO BADFORM;
48        /* END OF HEADING, OUTPUT CODE */
49        ENDSTA: IF HEADCNT<133 THEN DO; /* BLANK FILL ARRAY */
50            HEADCNT=HEADCNT+1; DO I=HEADCNT TO 133; HEADING(I)=' '; END;
51            PUT FILE (OBJECT) EDIT ('ON ENDPAGE (') (A(12));
52            IF FILEFLG THEN PUT FILE (OBJECT) EDIT (('FILENAM(I) DO I=1 TO
53            FILECNT)) (32 A(1));
54            ELSE PUT FILE (OBJECT) EDIT ('SYSPRINT') (A(8));
55            PUT FILE (OBJECT) EDIT (') PUT FILE (') (A(12));
56            IF FILEFLG THEN PUT FILE (OBJECT) EDIT (('FILENAM(I) DO I=1 TO
57            FILECNT)) (32 A(1));
58            ELSE PUT FILE (OBJECT) EDIT ('SYSPRINT') (A(8));
59            PUT FILE (OBJECT) EDIT (') EDIT (',',(HEADING(I) DO I=1 TO
60            133),',') (PAGE,A(133));') (A(9),133 A(1),A(17));
61            RETURN;
62        BADEND: CALL NEXTONE; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;

```

```

1      END;
2
3      /* AN EDIT BLOCK HAS BEEN FOUND, CLEAR FLAGS, READ ON.          */
4      EXECFLG=0; EDITFLG=0; FLAGFLG=0; LISTFLG=0;                      */
5      FORFLG=0; FILEFLG=0; PRALFLG=0; HEADFLG=0; MAXVAL=2;
6      FILEXX='SYSPRINT'; FILECNT=8; SUBFLG=0;
7      COLONE=0; COLCNT=73; /* START VALUES FOR NEXTONE */
8      NEXTSTA: CALL KEYWORD; /*GET KEYWORD */
9      IF EXECFLG THEN DO; /*BRANCH FOR EXECUTABLE/DECLARTV*/
10     IF (FLAG<=14)|| (FLAG>26) THEN GO TO PROCEXEC;
11     IF FLAG=26 THEN GO TO PROCESS;
12     IF FLAG=15 THEN DO; /*END STATEMENT, TEST FOR END */
13     CALL COLCHK;
14     CALL ENTITY; /*EDIT OR SUBEDIT */
15     IF WORD='EDIT' THEN DO; /*END EDIT, PROCESS & CONTINUE */
16     CALL COMPLT; GO TO LOOKNEXT; END;
17     IF WORD='SUBEDIT' THEN DO; /*END SUBEDIT,PROCESS & CONT*/
18     CALL ENDSUB; GO TO NEXTSTA; END;
19     PUT FILE(SYSPRINT) EDIT (' *** FORMAT OF END STATEMENT ',
20     'INVALID. END EDIT STATEMENT ASSUMED') (SKIP,A(29),A(35))
21     ; ERROR=1;
22     CALL COMPLT; GO TO LOOKNEXT;
23     END;
24     /* DECLARATIVE STATEMENT FOLLOWS EXECUTABLE STATEMENT(S) */
25     PUT FILE(SYSPRINT) EDIT (' *** ALL DECLARATIVE STATEMENTS ',
26     'MUST PRECEDE THE FIRST EXECUTABLE STATEMENT')
27     (SKIP,A(32),A(43)); ERROR=1;
28     FLAG=29; GO TO PROCESS;
29     END;
30
31     /* EXECUTABLE FLAG NOT SET, EXECUTABLE STATEMENT POSSIBLE */
32     ELSE DO;
33     IF ((FLAG>=14)&(FLAG<=25))||(FLAG=29) THEN GO TO PROCESS;
34     IF (EDITFLG=0)|| (FLAGFLG=0)|| (LISTFLG=0) THEN DO;
35     /* DECLARATIVE STATEMENTS NOT COMPLETE */
36     PUT FILE(SYSPRINT) EDIT (' *** THE DECLARATIVE SECTION ',
37     'MUST CONTAIN ONE EDIT, ONE FLAG AND ONE LIST OR ',
38     'PRINTNONE STATEMENT') (SKIP,A(29),A(48),A(19));
39     ERROR=1;
40     END;
41     EXECFLG=1; CALL EXECONE;
42     IF FLAG=26 THEN GO TO PROCESS;
43     END;
44
45     /* CALL SUBROUTINES TO CHECK COLUMN AND PROCESS STATEMENT */
46     PROCEXEC: CALL COLCHK;
47     PROCESS: IF FLAG= 1 THEN DO; CALL IFST; GO TO EX; END;
48     IF FLAG= 2 THEN DO; CALL ERST; GO TO EX; END;
49     IF FLAG= 3 THEN DO; CALL QUST; GO TO EX; END;
50     IF FLAG= 4 THEN DO; CALL ADST; GO TO EX; END;
51     IF FLAG= 5 THEN DO; CALL ELST; GO TO EX; END;
52     IF FLAG= 6 THEN DO; CALL COST; GO TO EX; END;
53     IF FLAG= 7 THEN DO; CALL LOST; GO TO EX; END;
54     IF FLAG= 8 THEN DO; CALL DOST; GO TO EX; END;
55     IF FLAG= 9 THEN DO; CALL GOST; GO TO EX; END;
56     IF FLAG=11 THEN DO; CALL DEST; GO TO EX; END;
57     IF FLAG=12 THEN DO; CALL CHST; GO TO EX; END;
58     IF FLAG=13 THEN DO; CALL PEST; GO TO EX; END;
59     IF FLAG=14 THEN DO; CALL STST; GO TO EX; END;
60     IF FLAG=16 THEN DO; CALL DCST; GO TO EX; END;

```



```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

```

IF FLAG=17 THEN DO; CALL EDST; GO TO EX; END;
IF FLAG=18 THEN DO; CALL FLST; GO TO EX; END;
IF FLAG=19 THEN DO; CALL MAST; GO TO EX; END;
IF FLAG=20 THEN DO; CALL LIST; GO TO EX; END;
IF FLAG=21 THEN DO; CALL PNST; GO TO EX; END;
IF FLAG=22 THEN DO; CALL FOST; GO TO EX; END;
IF FLAG=23 THEN DO; CALL FIST; GO TO EX; END;
IF FLAG=24 THEN DO; CALL HEST; GO TO EX; END;
IF FLAG=25 THEN DO; CALL PRST; GO TO EX; END;
IF FLAG=26 THEN DO; CALL LAST; GO TO EX; END;
IF FLAG=27 THEN DO; CALL AIST; GO TO EX; END;
IF FLAG=28 THEN DO; CALL I2ST; GO TO EX; END;
IF FLAG=29 THEN DO; CALL XXST; GO TO EX; END;
/* IF IT WAS EXECUTABLE THEN SET LIST OF XREF FLAGS */
EX: IF ALLOCATION(RECORD) THEN DO;
    IF (FLAG<=13) [(FLAG=27)] [(FLAG=28)] THEN DO;
        DO I=1 TO NITEM;
            IF RECFLAG(I)='1'B THEN DO;
                DO J=1 TO NITEM;
                    IF RECFLAG(J)='1'B THEN XREF(I,J)='1'B;
                END;
            END;
        END;
        DO I=1 TO NITEM; RECFLAG(I)='0'B; END;
    END;
    GO TO NEXTSTA;
END FRED;

```

```

1
2
3 FLST:  PROCEDURE;
4        /* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67
5           TO EDIT AND PROCESS THE FLAG STATEMENT */
6
7 DECLARE (ENDFLAG  FIXED BINARY(1),
8         ERROR     FIXED BINARY(1),
9         SYNTAX(32) PACKED CHARACTER(1),
10        SYNCNT    FIXED BINARY,
11        FLGWORD(32) PACKED CHARACTER(1),
12        FLGCNT    FIXED BINARY,
13        FLAGFLG    FIXED BINARY(1) ) STATIC EXTERNAL,
14        ALPHA     CHARACTER(29) INITIAL
15              ('ABCDEFGHIJKLMNOPQRSTUVWXYZ$@#') STATIC,
16        ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA;
17
18 /* THERE MUST BE ONE AND ONLY ONE FLAG STATEMENT */
19 IF FLAGFLG=1 THEN DO; /* PREVIOUS STATEMENT */
20   PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST FLAG',
21   ' STATEMENT') (SKIP,A(31),A(10));
22   ERROR=1; GO TO BADEND;
23 END;
24 FLAGFLG=1; /* SET THE FLAG */
25 CALL ENTITY;
26 IF ENDFLAG=1 THEN DO; /* STATEMENT CANNOT STOP HERE */
27   PUT FILE (SYSPRINT) EDIT (' *** NO FLAG IS SPECIFIED')
28   (SKIP,A(25));
29   ERROR=1; RETURN; END;
30
31 DO I=1 TO 29; /* FIRST CHARA MUST BE ALPHA */
32   IF SYNTAX(1)=ALPHABET(I) THEN GO TO VALID;
33 END;
34 PUT FILE (SYSPRINT) EDIT (' *** FLAG NAME MUST HAVE ',
35 'ALPHABETIC FIRST CHARACTER') (SKIP,A(25),A(26));
36 ERROR=1; GO TO BADEND;
37
38 VALID:  FLGWORD=SYNTAX; FLGCNT=SYNCNT; /*STORE WORD */
39 CALL ENTITY; /* THIS SHOULD BE END OF STATEMENT */
40 IF ENDFLAG=1 THEN RETURN;
41 PUT FILE (SYSPRINT) EDIT (' *** FLAG NAME IS NOT SINGLE WORD',
42 ' FORMAT') (SKIP,A(33),A(7));
43 ERROR=1;
44
45 BADEND: CALL ENTITY; IF ENDFLAG=1 THEN GO TO BADEND;
46 RETURN;
47
48 END FLST;
49
50

```



```

1  FOST:  PROCEDURE;
2          /* FRED PRE-PROCESSOR, DAVID MENDUS, 26/3/67
3          TO EDIT AND PROCESS A FORMAT STATEMENT */
4
5  DECLARE (ENDFLAG  FIXED BINARY(1),
6          SYNTAX(32) PACKED CHARACTER(1),
7          SYNCNT    FIXED BINARY,
8          ERROR     FIXED BINARY(1),
9          FORFLG    FIXED BINARY(1)          ) STATIC EXTERNAL,
10         BRCNT     FIXED BINARY;
11
12         /* SHOULD BE ONLY ONE FORMAT STATEMENT */
13         IF FORFLG THEN DO;
14             PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST ',
15             'FORMAT STATEMENT') (SKIP,A(27),A(16)); ERROR=1;
16             GO TO BADEND; END;
17         FORFLG=1; /* WRITE LABEL AND FIRMAT */
18         PUT FILE (OBJECT) EDIT (' Q #FORM: FORMAT ') (A(17));
19         CALL ENTITY; IF ENDFLAG THEN DO;
20             TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** FORMAT STATEMENT IS ',
21             'INCOMPLETE') (SKIP,A(25),A(10)); ERROR=1; RETURN;
22         END;
23         IF SYNTAX(1)='(' THEN DO;
24             PUT FILE (SYSPRINT) EDIT (' *** FORMAT DOES NOT START ',
25             'WITH (') (SKIP,A(27),A(6)); ERROR=1; GO TO BADEND;
26         END;
27         BRCNT=1; /* COUNT OF EXCESS ( OVER ) */
28         LOOP: PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
29             (X(1),32 A(1));
30         CALL ENTITY; IF ENDFLAG THEN DO;
31             IF BRCNT=0 THEN GO TO CLOSE; ELSE GO TO TOOSOON; END;
32             IF SYNTAX(1)='(' THEN DO;
33                 BRCNT=BRCNT+1; GO TO LOOP; END;
34             IF SYNTAX(1)=')' THEN DO;
35                 BRCNT=BRCNT-1; IF BRCNT<0 THEN DO;
36                     PUT FILE (SYSPRINT) EDIT (' *** RIGHT PARENTHESES EXCEED',
37                     'LEFT PARENTHESES') (SKIP,A(29),A(17)); ERROR=1;
38                     BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
39                 END;
40                 GO TO LOOP; END;
41             GO TO LOOP;
42         CLOSE: PUT FILE (OBJECT) EDIT (';') (A(1));
43         END FOST;

```

```

1 FRED: PROCEDURE OPTIONS (MAIN);
2 /* FRED PRE-PROCESSOR, DAVID MENDUS, 25/2/67
3 MAIN CONTROL ROUTINE. DOES LITTLE MORE THAN SELECT THE
4 APPROPRIATE SUBROUTINE TO PROCESS EACH STATEMENT AND UPDATE
5 THE CROSS-REFERENCE TABLE IF APPROPRIATE, */
6 DECLARE (ERROR FIXED BINARY(1),
7 COLONE FIXED BINARY,
8 COLCNT FIXED BINARY,
9 ENDPROG FIXED BINARY(1),
10 EDITFLG FIXED BINARY(1),
11 FLAGFLG FIXED BINARY(1),
12 FORFLG FIXED BINARY(1),
13 FILEFLG FIXED BINARY(1),
14 FILENAM(32) PACKED CHARACTER(1),
15 FILECNT FIXED BINARY,
16 PRALFLG FIXED BINARY(1),
17 HEADFLG FIXED BINARY(1),
18 MAXVAL FIXED BINARY,
19 LISTFLG FIXED BINARY(1),
20 SUBFLG FIXED BINARY(1),
21 FLAG FIXED BINARY,
22 SYNTAX(32) PACKED CHARACTER(1),
23 NITEM FIXED BINARY) STATIC EXTERNAL,
24 WORD CHARACTER(32) DEFINED SYNTAX,
25
26 (RECFLAG(NITEM) BIT(1),
27 RECORD(NITEM) PACKED CHARACTER(12),
28 XREF(NITEM,NITEM) BIT(1) ) CONTROLLED EXTERNAL,
29
30 FILEXX CHARACTER(32) DEFINED FILENAM,
31 EXECFLG FIXED BINARY(1) STATIC;
32
33 /* PAGE HEADINGS */
34 ON ENDPAGE (SYSPRINT) BEGIN;
35 IPAGE=IPAGE+1;
36 PUT FILE (SYSPRINT) EDIT (' FRED PRE-PROCESSOR ***','PAGE',
37 IPAGE) (PAGE,A(23),X(89),A(4),F(4,0));
38 PUT FILE (SYSPRINT) SKIP;
39 END;
40 IPAGE=1;
41 PUT FILE (SYSPRINT) EDIT (' FRED PRE-PROCESSOR ***','PAGE',
42 IPAGE) (PAGE,A(23),X(89),A(4),F(4,0));
43 PUT FILE (SYSPRINT) SKIP;
44
45 /* READ THROUGH DECK UNTIL START EDIT STATEMENT, PROCESS IT. */
46 ERROR=0; ENDPROG=0;
47 LOOKNEXT:CALL STARTER; /* TO LOOK FOR NEXT 'START EDIT' STATEMENT*/
48 IF ENDPROG THEN DO;
49 IF ERROR THEN DO;
50 PUT FILE(SYSPRINT) EDIT (' END OF THIS COMPILATION, ',
51 'ERRORS WERE NOTED AS SIGNALLED ABOVE. ',' DO NOT
52 'ATTEMPT TO RUN THIS PROGRAM UNTIL THESE ERRORS HAVE ',
53 'BEEN REMOVED.') (SKIP(3),A(26),A(38),SKIP,A(8),A(52),
54 A(13));
55 RETURN; END; /*'STOP' AFTER TESTING */
56 ELSE DO;
57 PUT FILE(SYSPRINT) EDIT (' END OF THIS COMPILATION, ',
58 'NO ERRORS WERE NOTED IN THE EDITING STATEMENTS')
59 (SKIP(3),A(26),A(46));
60 RETURN; END; /*'STOP' AFTER TESTING */

```



```

1
2
3 FIST: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 24/3/67
5 TO EDIT AND PROCESS A FILE STATEMENT */
6
7 DECLARE (ENDFLAG FIXED BINARY(1),
8 SYNTAX(32) PACKED CHARACTER(1),
9 SYNCNT FIXED BINARY,
10 ERROR FIXED BINARY(1),
11 FILEFLG FIXED BINARY(1),
12 HEADFLG FIXED BINARY(1),
13 FILENAM(32) PACKED CHARACTER(1),
14 FILECNT FIXED BINARY ) STATIC EXTERNAL,
15
16 ALPHA CHARACTER(29) STATIC INITIAL
17 ('ABCDEFGHIJKLMNOPQRSTUVWXYZ#@$',),
18 ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA;
19
20 IF HEADFLG THEN DO; /* MUST PRECEDE HEADING STATEMENT */
21 PUT FILE (SYSPRINT) EDIT (' *** THIS DOES NOT PRECEDE ',
22 'THE HEADING STATEMENT') (SKIP,A(27),A(21));
23 ERROR=1; GO TO BADEND; END;
24 IF FILEFLG THEN DO; /* ONLY ONE FILE STATEMENT ALLOWED */
25 PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST FILE',
26 'STATEMENT') (SKIP,A(31),A(10)); ERROR=1;
27 BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
28 END;
29 FILEFLG=1; /* SET FLAG, READ FILE NAME */
30 CALL ENTITY;
31 IF ENDFLAG THEN DO;
32 PUT FILE (SYSPRINT) EDIT (' *** FILE STATEMENT IS ',
33 'INCOMPLETE') (SKIP,A(23),A(10)); ERROR=1; RETURN;
34 END;
35
36 /* CHECK FIRST CHARACTER FOR ALPHA AND STORE NAME */
37 DO I=1 TO 29; IF SYNTAX(1)=ALPHABET(I) THEN GO TO OK; END;
38 PUT FILE (SYSPRINT) EDIT (' *** FILE NAME GIVEN IS INVALID')
39 (SKIP,A(31)); ERROR=1; GO TO BADEND;
40 FILENAM=SYNTAX; FILECNT=SYNCNT; /* SHOULD BE NO MORE */
41 CALL ENTITY; IF ENDFLAG THEN RETURN;
42 PUT FILE (SYSPRINT) EDIT (' *** FILE STATEMENT FORMAT INVALID'
43 ) (SKIP,A(34)); ERROR=1; GO TO BADEND;
44
45 END FIST;
46
47
48
49
50

```

```

1
2
3 NEXTRHS: CALL ENTITY; IF ENDFLAG THEN GO TO CHECKEND;
4       PUT FILE (OBJECT) EDIT ((STORE(I) DO I=1 TO STRCNT))
5       (X(1),32 A(1));
6       STORE=SYNTAX; STRCNT=SYNCNT; GO TO NEXTRHS;
7
8 CHECKEND: IF (STORE(1)='*') & (STRCNT=1) THEN DO;
9       PUT FILE (OBJECT) EDIT (';') (A(2)); RETURN; END;
10      PUT FILE (SYSPRINT) EDIT (' *** ILLEGAL END TO STATEMENT')
11      (SKIP,A(29));
12      ERROR=1; RETURN;
13
14 FINISH: CALL ENTITY; IF ENDFLAG THEN RETURN;
15      PUT FILE (SYSPRINT) EDIT (' *** ILLEGAL END TO STATEMENT')
16      (SKIP,A(29));
17
18 BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
19
20 END ERST;
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```



```

1
2
3 EXECONE: PROCEDURE;
4      /* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
5         TO WRITE FIRST EXECUTABLE STATEMENTS OF OBJECT CODE */
6
7 DECLARE NITEM      FIXED BINARY STATIC EXTERNAL,
8          RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL;
9
10      PUT FILE (OBJECT) EDIT ('Q_#FATAL,Q_#QUERY,Q_#ADJUST,',
11      'Q_#DIGCT=0;') (A(28),A(11));
12      IF HALALLOCATION(RECORD) THEN RETURN;
13      DO I=1 TO NITEM;
14          PUT FILE (OBJECT) EDIT ('Q_#',RECORD(I),'=',RECORD(I),';')
15          (A(3),A(12),A(1),A(12),A(1));
16      END;
17      RETURN;
18 END EXECONE;
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

```

1
2
3 ERST:  PROCEDURE; /* ENTRY QUST, ADCON */
4         /* FRED PRE-PROCESSOR, DAVID MENDUS, 18/3/67
5         TO PROCESS ERROR OR QUERY STATEMENTS INCLUDING RRROR FLAG
6         SETTING AND DIAGNOSTIC ACTION, ALSO COMPLETES ADJUST
7         STATEMENT DECODING */
8
9 DECLARE (ENDFLAG  FIXED BINARY (1),
10         SYNTAX(32) PACKED CHARACTER (1),
11         SYNCNT   FIXED BINARY,
12         ERROR    FIXED BINARY(1),
13         LISTFLG  FIXED BINARY(1) ) STATIC EXTERNAL,
14
15         (STORE(32) PACKED CHARACTER(1),
16         STRCNT   FIXED BINARY ) STATIC;
17
18         /* WRITE CODE TO SET ERROR FLAG */
19         PUT FILE (OBJECT) EDIT (' Q_#FATAL=1;') (A(12)); GO TO READON;
20
21         /* WRITE CODE TO SET QUERY FLAG */
22 QUST:  ENTRY;
23 ADCON: ENTRY; /* TO COMPLETE DECODING ADJUST STATEMENT */
24
25 READON: CALL ENTITY; /* NEXT UNIT SELECTS DIAGNOSTIC OPTION */
26 IF ENDFLAG THEN DO; /* STATEMENT CANNOT END HERE */
27 ENDSOON: PUT FILE (SYSPRINT) EDIT (' *** STATEMENT INCOMPLETE' )
28         (SKIP,A(25)); ERROR=1; RETURN; END;
29
30 IF (SYNCNT=1)&(SYNTAX(1)='*') THEN GO TO FLAGIT;
31 IF LISTFLG=-1 THEN DO; /* ILLEGAL LISTING OPTION */
32 PUT FILE (SYSPRINT) EDIT (' *** ERROR MESSAGE IS ILLEGAL ',
33 'AS PRINTNONE OPTION IS SELECTED') (SKIP,A(30),A(31));
34 ERROR=1; GO TO BADEND; END;
35 PUT FILE (OBJECT) EDIT (' IF Q_#DIGCT=40 THEN Q_#DIGCT',
36 '=Q_#DIGCT+1;') (A(30),A(12));
37 IF SYNTAX(1)='*' THEN DO; /* CHARACTER STRING CONSTANT */
38 PUT FILE (OBJECT) EDIT (' Q_#DIAG(Q_#DIGCT)=', (SYNTAX(1)
39 DO I=1 TO SYNCNT),';') (A(19),33 A(1)); END;
40 ELSE DO; /* CHARACTER STRING VARIABLE */
41 PUT FILE (OBJECT) EDIT (' Q_#DIAG(Q_#DIGCT)=', (SYNTAX(1)
42 DO I=1 TO SYNCNT),';') (A(19),33A(1)); END;
43 GO TO FINISH;
44
45 FLAGIT: IF LISTFLG=1 THEN DO; /* MUST BE AN ERROR MESSAGE */
46 PUT FILE (SYSPRINT) EDIT (' *** AN ERROR MESSAGE MUST BE ',
47 'GIVEN AS LISTING OPTION WAS SELECTED') (SKIP,A(30),A(36));
48 ERROR=1; GO TO BADEND; END;
49
50 READLHS: CALL ENTITY; IF ENDFLAG THEN GO TO ENDSOON;
51 WRITELHS: PUT FILE (OBJECT) EDIT ((SYNTAX(1) DO I=1 TO SYNCNT))
52 (X(1),32 A(1)); /* FIRST PART OF L.H.S. */
53 WRITEQ: CALL ENTITY; IF ENDFLAG THEN GO TO ENDSOON;
54 IF SYNTAX(1)='=' THEN GO TO WRITERHS;
55 IF SYNTAX(1)='(' THEN DO; /* BRACKET VALID HERE */
56 PUT FILE (OBJECT) EDIT ('(') (X(1),A(1));
57 CALL BRACKET; GO TO READLHS; END;
58 PUT FILE (SYSPRINT) EDIT (' *** INVALID FORM OF ASSIGNMENT' )
59 (SKIP,A(31)); ERROR=1; GO TO BADEND;
60
61 WRITERHS: PUT FILE (OBJECT) EDIT ('=') (X(1),A(1)); STORE=' ';
62 STRCNT=1;

```

```

1
2
3      RETURN;
4  END;
5  ELSE DO;          /* ANOTHER CHARACTER, CHECK VALID LENGTH */
6  IF SYNCNT<32 THEN DO; /* STORE NEXT CHARACTER */
7      SYNCNT=SYNCNT+1; SYNTAX(SYNCNT)=CHARA;
8      GO TO ALPHA; END;
9  ELSE DO;          /* TOO LONG */
10     PUT FILE (SYSPRINT) EDIT (' *** IDENTIFIER EXCEEDS',
11     '32 CHARACTERS') (SKIP,A(23),A(13));
12     ERROR=1;      /*SKIP CHARACTERS TO NEXT BLANK */
13     DO WHILE (INWORD=1);
14     CALL NEXTONE; END;
15     RETURN;
16 END;
17 END ENTITY;
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```



```

1
2
3 ENTITY: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 20/2/67
5 ROUTINE CALLED AS PART OF THE RECOGNISER TO ASSEMBLE THE
6 NEXT WORD OR STRING OF SPECIAL CHARACTERS. UPDATES THE
7 CROSS REFERENCE TABLE IF APPROPRIATE. */
8
9 DECLARE (ENDFLAG FIXED BINARY(1),
10 INWORD FIXED BINARY(1),
11 REPEAT FIXED BINARY(1),
12 ERROR FIXED BINARY(1),
13 CHARA CHARACTER(1),
14
15 SYNTAX(32) PACKED CHARACTER(1),
16 SYNCNT FIXED BINARY(1) STATIC EXTERNAL,
17 WORD CHARACTER(32) DEFINED SYNTAX,
18
19 (RECORD(NITEM) PACKED CHARACTER(12),
20 RECFLAG(NITEM) BIT(1) CONTROLLED EXTERNAL,
21 NITEM FIXED BINARY(1) STATIC EXTERNAL;
22 /* GET FIRST CHARACTER, IF END OF STATEMENT THEN RETURN */
23 WORD=''; SYNCNT=0;
24
25 RESTART: CALL NEXTONE;
26 IF ENDFLAG THEN RETURN;
27 IF INWORD=0 THEN GO TO RESTART;
28 SYNCNT=1; SYNTAX(1)=CHARA;
29 IF INWORD=-1 THEN GO TO ALPHA;
30
31 /* SPECIAL CHARACTER, RETURN CHARACTER OR PAIR */
32 IF (CHARA='>') || (CHARA='<') || (CHARA='~') THEN DO;
33 CALL NEXTONE; /* >,<,- ARE VALID */
34 IF CHARA='~' THEN DO;
35 SYNCNT=2; SYNTAX(2)=CHARA; RETURN; END;
36 ELSE DO;
37 REPEAT=1; RETURN; END;
38 END;
39 IF CHARA='*' THEN DO; /* ** IS VALID */
40 CALL NEXTONE;
41 IF CHARA='*' THEN DO;
42 SYNCNT=2; SYNTAX(2)=CHARA; RETURN; END;
43 ELSE DO;
44 REPEAT=1; RETURN; END;
45 END;
46 IF CHARA='|' THEN DO; /* || IS VALID */
47 CALL NEXTONE;
48 IF CHARA='|' THEN DO;
49 SYNCNT=2; SYNTAX(2)=CHARA; RETURN; END;
50 ELSE DO;
51 REPEAT=1; RETURN; END;
52 END;
53 RETURN;
54
55 /* ALPHA CHARACTER, BUILD UP STRING OF CHARACTERS */
56 ALPHA: CALL NEXTONE;
57 IF INWORD=0 THEN DO; /*END OF CHARACTER STRING */
58 IF ALLOCATION(RECORD) THEN DO;
59 DO I=1 TO NITEM; /*LOOK FOR THE NAME OF AN ITEM */
60 IF RECORD(I)=WORD THEN DO;
61 RECFLAG(I)='1'B; RETURN; END;
62 END;
63 END;

```

```

1
2
3      ENDSUB:  PROCEDURE;
4      /* FRED PRE/PROCESSOR, DAVID MENDUS, 24/3/67
5      TO PROCESS AN END SUBEDIT STATEMENT */
6
7      DECLARE (ENDFLAG  FIXED BINARY(1),
8      SYNTAX(32) PACKED CHARACTER(1),
9      SYNCNT  FIXED BINARY,
10     ERROR   FIXED BINARY(1),
11     SUBFLG  FIXED BINARY(1),
12     SUBNAME(32) PACKED CHARACTER(1),
13     SUBCNT  FIXED BINARY ) STATIC EXTERNAL,
14     WORD    CHARACTER(32) DEFINED SYNTAX,
15     WORDX   CHARACTER(32) DEFINED SUBNAME;
16
17     IF SUBFLG=0 THEN DO; /* OUTSIDE SUBEDIT, ERROR */
18     PUT FILE (SYSPRINT) EDIT (' *** THIS STATEMENT DOES NOT ',
19     'OCCUR IN A SUBEDIT') (SKIP,A(29),A(18)); ERROR=1;
20     GO TO BADEND; END;
21     SUBFLG=0; PUT FILE (OBJECT) EDIT (' END') (A(4));
22
23     CALL ENTITY; /* SUBEDIT MAY OR MAY NOT BE NAMED */
24     IF ENDFLAG=0 THEN DO; /* CHECK NAME IF GIVEN */
25     IF WORD=WORDX THEN DO;
26     PUT FILE (SYSPRINT) EDIT (' *** NAME GIVEN IN END ',
27     'STATEMENT IS NOT NAME OF SUBEDIT') (SKIP,A(23),A(32));
28     ERROR=1; GO TO BADEND;
29     END;
30     PUT FILE (OBJECT) EDIT ((SUBNAME(I) DO I=1 TO SUBCNT),';')
31     (X(1),33 A(1));
32     CALL ENTITY; /* SHOULD BE END OF STATEMENT */
33     IF ENDFLAG THEN RETURN;
34     PUT FILE (SYSPRINT) EDIT (' *** END SUBEDIT FORMAT INVALID')
35     (SKIP,A(31)); ERROR=1;
36     BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
37     END ENDSUB;
38
39
40
41
42
43
44
45
46
47
48
49
50

```

```

1
2
3      ELST:  PROCEDURE;
4              /* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67
5                 TO PROCESS AN ELSE-STATEMENT, THE IF-BLOCKING HAS
6                 ALREADY BEEN DONE BY COLCHK. */
7
8      DECLARE (ENDFLAG  FIXED BINARY (1),
9                ERROR    FIXED BINARY (1) ) STATIC EXTERNAL;
10
11      CALL ENTITY; IF ENDFLAG=1 THEN GO TO WRITE;
12      PUT FILE (SYSPRINT) EDIT (' *** ELSE STATEMENT CONTAINS ',
13      'ILLEGAL WORDS') (SKIP,A(29),A(13));
14      ERROR=1;
15      READ:   CALL ENTITY; IF ENDFLAG=1 THEN GO TO READ;
16
17      WRITE:  PUT FILE (OBJECT) EDIT (' END;') (A(5));
18      RETURN;
19      END ELST;
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```


1
2
3 ALLOCATE RECORD; ALLOCATE RECFLAG; ALLOCATE XREF;
4 DO I=1 TO NITEM; RECORD(I)=RECARRAY(I); END;
5 DO I=1 TO NITEM; RECFLAG(I)='0'B;
6 DO J=1 TO NITEM; XREF(I,J)='0'B; END;
7 END;
8 RETURN;

9 BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
10 END EDST;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

EDST: PROCEDURE;
/* FRED PRE-PROCESSOR, DAVID MENDUS, 26/3/67
TO EDIT AND PROCESS THE EDIT STATEMENT. WRITES DECLARE
STATEMENT FOR COPY AND LISTS ITEM NAMES FOR EDITS AND
X-REF TABLES. */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCNT FIXED BINARY,
ERROR FIXED BINARY(1),
EDITFLG FIXED BINARY(1),
NITEM FIXED BINARY) STATIC EXTERNAL,
WORD CHARACTER(32) DEFINED SYNTAX,

(RECFLAG(NITEM) BIT(1),
RECORD(NITEM) PACKED CHARACTER(12),
XREF(NITEM,NITEM) BIT(1)) CONTROLLED EXTERNAL,

RECARRAY(100) PACKED CHARACTER(12),
ALPHA CHARACTER(29) STATIC INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ#@\$'),
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA,
NUMB CHARACTER(10) STATIC INITIAL ('0123456789'),
NUMBERS(10) PACKED CHARACTER(1) DEFINED NUMB;

IF EDITFLG THEN DO; /* CAN ONLY BE ONE EDIT STATEMENT */
PUT FILE (SYSPRINT) EDIT (' *** THIS IS NOT THE FIRST ',
'EDIT STATEMENT') (SKIP,A(27),A(14)); ERROR=1;
GO TO BADEND; END;
EDITFLG=1; NITEM=0;
PUT FILE (OBJECT) EDIT (' DECLARE') (A(8));

LOOP1: CALL ENTITY; IF ENDFLAG THEN DO;
PUT FILE (SYSPRINT) EDIT (' *** EDIT STATEMENT IS ',
'INCOMPLETE') (SKIP,A(23),A(10)); ERROR=1; RETURN;
END; /* CHECK FIRST CHARA FOR ALPHA. IF SO ASSUME NAME */
DO I=1 TO 29; IF SYNTAX(1)=ALPHABET(I) THEN GO TO NAME; END;
/* IF NUMERIC MAY BE LEVEL, WRITE AND LOOK FOR NAME */
DO I=1 TO 10; IF SYNTAX(1)=NUMBERS(I) THEN DO;
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1)); GO TO LOOP1; END; END;
GO TO WRITEIT; /* NOT NAME NOR NUMBER */

NAME: IF SYNCNT>12 THEN DO;
PUT FILE (SYSPRINT) EDIT (' *** ITEM NAME EXCEEDS 12 ',
'CHARACTERS') (SKIP,A(26),A(10)); ERROR=1; GO TO BADEND;
END;
NITEM=NITEM+1; RECARRAY(NITEM)=WORD; /* STORE NAME */
PUT FILE (OBJECT) EDIT (' Q_#', (SYNTAX(I) DO I=1 TO SYNCNT))
(A(4),32 A(1));

/* NEXT UNIT CANNOT BE ITEM NAME */
LOOP2: CALL ENTITY; IF ENDFLAG THEN GO TO TIDYUP;
WRITEIT: PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1)); /* IF , OR (THEN NEXT MAY BE NAME */
IF (SYNTAX(1)=',') || (SYNTAX(1)='(') THEN GO TO LOOP1;
ELSE GO TO LOOP2;

/* ALL DONE, TIDY UP AND ALLOCATE ARRAYS */
TIDYUP: PUT FILE (OBJECT) EDIT (';') (A(1));

```

1
2      DOST:  PROCEDURE;
3             /* FRED PRE-PROCESSOR, DAVID MENDUS, 9/3/67
4             TO PROCESS AN EMBEDDED DO GROUP.                                */
5
6      DECLARE (ENDFLAG  FIXED BINARY(1),
7               SYNTAX(32) PACKED CHARACTER(1),
8               SYNCNT   FIXED BINARY(1),
9               ERROR     FIXED BINARY(1)      ) STATIC EXTERNAL,
10             (ONEFLAG   FIXED BINARY(1),
11             BUFFER(32) PACKED CHARACTER(1),
12             BUFCT      FIXED BINARY(1)      ) STATIC;
13
14      DO I=1 TO 32; BUFFER(I)=' '; END;
15      ONEFLAG=1;
16      NEXTUNIT: CALL ENTITY;                                /*GET NEXT UNIT */
17      IF ENDFLAG THEN DO;                                    /*IF ENDFLAG SET, THEN LAST -*/
18          IF (BUFFER(1)='E') &                                /* UNIT MUST BE END, ELSE ERROR*/
19              (BUFFER(2)='N') & (BUFFER(3)='D') & (BUFCT=3) THEN RETURN;
20          PUT FILE (SYSPRINT) EDIT (' *** THIS DO-STATEMENT DOES',
21          'NOT HAVE VALID END') (SKIP,A(27),A(18));
22          ERROR=1; RETURN;
23      END;
24
25      IF ONEFLAG THEN DO;                                    /* NO OUTPUT IF ONEFLAG IS SET*/
26          ONEFLAG=0; END;
27      ELSE DO;
28          PUT FILE (OBJECT) EDIT ((BUFFER(I) DO I=1 TO BUFCT))
29          (X(1),32 A(1)); END;
30
31      BUFFER=SYNTAX; BUFCT=SYNCNT; /*STORE THIS UNIT FOR NEXT */
32      GO TO NEXTUNIT;
33
34      END DOST;

```



```

1
2      IF BRSTC THEN DO;
3          PUT FILE (OBJECT) EDIT ((BRST(I) DO I=1 TO BRSTC))
4              (X(1),64 A(1)); END;
5      PUT FILE (OBJECT) EDIT ('Q_#',(WORDST(I) DO I=1 TO WORDSTC))
6          (A(4),32 A(1));
7      IF BRSTC THEN DO;
8          PUT FILE (OBJECT) EDIT ((BRST(I) DO I=1 TO BRSTC))
9              (X(1),64 A(1)); END;
10     PUT FILE (OBJECT) EDIT ('=') (A(1));
11
12     /* PROCESS R.H.S. */
13     CALL ENTITY; IF ENDFLAG=0 THEN GO TO WRITERHS;
14     TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** THIS STATEMENT IS INCOMPLETE')
15         (SKIP,A(33));
16     ERROR=1; RETURN;
17
18     WRITERHS: PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
19         (X(1),32 A(1));
20     CALL ENTITY; IF ENDFLAG=0 THEN GO TO WRITERHS;
21     PUT FILE (OBJECT) EDIT (' ;') (A(2));
22     RETURN;
23
24     BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
25     END DEST;
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

CHST:
DEST:

PROCEDURE;

/* FRED PRE-PROCESSOR, DAVID MENDUS, 19/3/67
TO EDIT AND PROCESS A DEFINE OR CHANGE STATEMENT. ACTION
IS IDENTICAL FOR BOTH */

DECLARE (ENDFLAG FIXED BINARY (1),
SYNTAX(32) PACKED CHARACTER (1) ,
SYNCNT FIXED BINARY,
ERROR FIXED BINARY (1),
NITEM FIXED BINARY) STATIC EXTERNAL,
RECORD(NITEM) PACKED CHARACTER (12) CONTROLLED EXTERNAL,
WORD CHARACTER (32) DEFINED SYNTAX,

(BRCNT FIXED BINARY,
WORDST(32) PACKED CHARACTER (1),
BRST(64) PACKED CHARACTER (1),
WORDSTC FIXED BINARY,
BRSTC FIXED BINARY) STATIC;

/* READ VARIABLE NAME OF LHS AND CHECK LEGALITY */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF ALLOCATION(RECORD) THEN DO;
DO I=1 TO NITEM;
IF WORD=RECORD(I) THEN GO TO WORDOK;
END; /* IF WORD CANNOT BE FOUND THEN ERROR */
PUT FILE (SYSPRINT) EDIT (' *** NAME GIVEN HAS NOT BEEN ',
'DECLARED AS PART OF THE RECORD') (SKIP,A(29),A(30));
ERROR=1; GO TO BADEND;
END;

WORDOK: /* READ TO = , LHS MAY INCLUDE PARENTHESED EXPRESSION */
WORDST=SYNTAX; WORDSTC=SYNCNT; /* STORE NAME */
BRSTC=0;

READEQ: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(1)='(' THEN DO; /* READ PARENTHESED EXP */

BRST(1)='('; BRSTC=1; BRCNT=1;
READBRAC: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF BRCNT+SYNCNT>64 THEN DO;
PUT FILE (SYSPRINT) EDIT (' *** ARRAY SUBSCRIPT LENGTH ',
'EXCEEDS 64 CHARACTERS ALLOWED BY COMPILER')
(SKIP,A(28),A(41));
ERROR=1; GO TO BADEND;

END;
DO I=1 TO SYNCNT; /* SAVE NEW UNIT IN STRING */
BRST(BRCNT+I)=SYNTAX(I);
END;
IF SYNTAX(1)=')' THEN DO;
BRCNT=BRCNT-1; IF BRCNT=0 THEN GO TO READEQ; END;
IF SYNTAX(1)='(' THEN BRCNT=BRCNT+1;
GO TO READBRAC;

END;

IF SYNTAX(1)='=' THEN GO TO EQUALS; /* MUST BE (OR = */
PUT FILE (SYSPRINT) EDIT (' *** ASSIGNMENT FORMAT INVALID')
(SKIP,A(30));
ERROR=1; GO TO BADEND;

EQUALS: PUT FILE (OBJECT) EDIT ((WORDST(I) DO I=1 TO WORDSTC))
(X(1),32 A(1));

```

1
2      DCST:  PROCEDURE;
3              /* FRED PRE-PROCESSOR, DAVID MENDUS, 18/3/67
4              TO PROCESS A DECLARE STATEMENT */
5
6      DECLARE (ENDFLAG FIXED BINARY (1),
7              SYNTAX(32) PACKED CHARACTER (1),
8              SYNCNT  FIXED BINARY,
9              ERROR    FIXED BINARY ) STATIC EXTERNAL;
10
11      GETNEXT: PUT FILE (OBJECT) EDIT (' DECLARE') (A(8));
12              CALL ENTITY;
13              IF ENDFLAG THEN DO;
14                  PUT FILE (OBJECT) EDIT (' ;') (A(2)); RETURN; END;
15              IF SYNTAX(1)='(' THEN DO;
16                  PUT FILE (OBJECT) EDIT (' (') (A(2));
17                  CALL BRACKET; GO TO GETNEXT; END;
18              ELSE DO;
19                  PUT FILE (OBJECT) EDIT ((SYNTAX(1) DO I=1 TO SYNCNT))
20                  (X(1),32 A(1)); GO TO GETNEXT; END;
21
22      END DCST;
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```



```

1
2
3 COST:  PROCEDURE;
4         /*FRED PRE-PROCESSOR, DAVID MENDUS, 26/2/67
5         ROUTINE TO PROCESS A CONTINUE OR A ** STATEMENT. NOTE THAT
6         THESE STATEMENTS DO NOT GENERATE ANY OBJECT CODE.          */
7 DECLARE (ERROR      FIXED BINARY (1),
8         ENDFLAG     FIXED BINARY(1)      ) STATIC EXTERNAL;
9
10        /* NO OTHER NON-BLANK CHARACTERS SHPULD BE PRESENT.          */
11        CALL ENTITY;
12        IF ENDFLAG=1 THEN RETURN;
13        PUT FILE(SYSPRINT) EDIT (' *** EXCESS CHARACTERS IN THIS ',
14        'CONTINUE STATEMENT HAVE BEEN DELETED') (SKIP,A(31),A(36));
15        ERROR=1;
16        DO WHILE (ENDFLAG/=1);
17            CALL ENTITY;
18        END;
19        RETURN;
20
21 END COST;
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

FREE RECORD; FREE XREF; FREE RECFLAG;
END COMPLT;

UNICO-FANFID-BURNING FORM

265563

```

1      PUT FILE (OBJECT) EDIT (' =1;') (A(4));
2      IF LISTFLG=-1 THEN DO;
3          PUT FILE (OBJECT) EDIT (' PUT FILE (',(FILENAM(I) DO I=1 TO
4              FILECNT)) (A(11),32 A(1));
5          PUT FILE (OBJECT) EDIT ('') EDIT (' RECORD ADJUSTED') ('',
6              'SKIP(2),A(16));') (A(29),A(15));
7      END;
8
9      PUT FILE (OBJECT) EDIT (' GO TO Q_#COMPLT; END;') (A(22));
10     /* DEAL WITH VALID RECORD */
11     PUT FILE (OBJECT) EDIT (('FLGWORD(I) DO I=1 TO FLGCNT))
12         (32 A(1));
13     PUT FILE (OBJECT) EDIT ('=0; Q_#COMPLT;') (A(14));
14
15     /* WRITE CODE TO WRITE ERROR LISTINGS */
16     IF LISTFLG=-1 THEN GO TO NOCODE;
17     IF PRALFLG THEN PUT FILE (OBJECT) EDIT (' DO;') (A(4));
18     ELSE PUT FILE (OBJECT) EDIT ('IF (Q_#FATAL)[(Q_#QUERY)]',
19         '(Q_#ADJUST) THEN DO;') (A(25),A(20));
20     PUT FILE (OBJECT) EDIT ('PUT FILE (',(FILENAM(I) DO I=1 TO
21         FILECNT)) (A(10),32 A(1));
22     IF FORFLG THEN PUT FILE (OBJECT) EDIT ('') EDIT ('') (A(8));
23     ELSE PUT FILE (OBJECT) EDIT ('') LIST ('') (A(8));
24     PUT FILE (OBJECT) EDIT (('LISTSTR(I) DO I=1 TO LISTCNT),')
25         (2001 A(1));
26     IF FORFLG THEN PUT FILE (OBJECT) EDIT ('(R(Q_#FORM))')
27         (A(12));
28     PUT FILE (OBJECT) EDIT ('; END;') (A(6));
29     PUT FILE (OBJECT) EDIT (' IF Q_#DIGCT=-0 THEN DO;') (A(24));
30     PUT FILE (OBJECT) EDIT (' PUT FILE (',(FILENAM(I) DO I=1 TO
31         FILECNT),')') (A(11),33 A(1));
32     PUT FILE (OBJECT) EDIT (' EDIT ((Q_#DIAG(IQ_#) DO IQ_#=1 ',
33         'TO Q_#DIGCT)) (SKIP,A(40));') (A(32),A(27));
34     PUT FILE (OBJECT) EDIT (' END;') (A(5));
35
36     /* CHECK NAME OF EDIT IS CORRECT */
37     NOCODE: CALL ENTITY; IF ENDFLAG THEN GO TO NONAME;
38     IF WORD=EDITNAM THEN DO;
39         PUT FILE (SYSPRINT) EDIT (' *** NAME GIVEN CANNOT BE ',
40             'RECOGNISED') (SKIP,A(26),A(10)); ERROR=1; END;
41     NONAME: PUT FILE (OBJECT) EDIT (' END ',EDITNAM,';') (A(5),A(7),A(1));
42     /* IF CROSS REFERENCE TABLE WAS FILLED THEN WRITE IT */
43     IF ALLOCATION(RECORD) THEN DO;
44         PUT FILE (SYSPRINT) EDIT (' LOGICAL CROSS-REFERENCE TABLE ',
45             'FOR THIS EDIT IS') (SKIP(60),A(31),A(16));
46         DO I=1 TO NITEM;
47             PUT FILE (SYSPRINT) EDIT (RECORD(I),' ****') (SKIP,A(12),
48                 A(5));
49             J=0; /* COUNT OF NAMES ON LINE */
50             DO K=1 TO NITEM;
51                 IF XREF(I,K)=1'B THEN DO;
52                     J=J+1; IF J=8 THEN DO;
53                         PUT FILE (SYSPRINT) EDIT (' ') (SKIP,X(16),A(1));
54                         J=1;
55                     END;
56                     PUT FILE (SYSPRINT) EDIT (RECORD(K)) (X(1),A(12));
57                 END;
58             END;
59         END;
60     END;
61     PUT FILE (SYSPRINT) SKIP(60);

```


COMPLT: PROCEDURE;
/* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
TO COMPLETE THE OBJECT CODE OF ONE EDIT */

```
DECLARE (ENDFLAG    FIXED BINARY(1),
        SYNTAX(32)  PACKED CHARACTER(1),
        SYNCNT     FIXED BINARY,
        ERROR      FIXED BINARY(1),
        LISTFLG    FIXED BINARY(1),
        LISTCNT    FIXED BINARY,
        FORFLG     FIXED BINARY(1),
        PRALFLG    FIXED BINARY(1),
        MAXVAL     FIXED BINARY,
        FILENAM(32) PACKED CHARACTER(1),
        FILECNT    FIXED BINARY,
        NITEM      FIXED BINARY,
        FLGWORD(32) PACKED CHARACTER(1),
        FLGCNT     FIXED BINARY,
        EDITNAM    CHARACTER(7)          ) STATIC EXTERNAL,
        RECORD(NITEM) PACKED CHARACTER(12) EXTERNAL CONTROLLED,
        XREF(NITEM,NITEM) BIT(1) CONTROLLED EXTERNAL,
        RECFLAG(NITEM) BIT(1) CONTROLLED EXTERNAL,
        LISTSTR(LISTCNT) PACKED CHARACTER(1) EXTERNAL CONTROLLED,
        WORD      CHARACTER(32) DEFINED SYNTAX;

/* WRITE CODE TO DEAL WITH FATAL ERROR RECORDS */
PUT FILE (OBJECT) EDIT (' IF Q_#ADJUST>',MAXVAL,' THEN ',
    'Q_#FATAL=1;') (A(14),F(1,0),A(6),A(11));
PUT FILE (OBJECT) EDIT (' IF Q_#FATAL THEN DO; ',(FLGWORD(I)
    DO I=1 TO FLGCNT)) (A(22),32 A(1));
PUT FILE (OBJECT) EDIT (' =3;') (A(4));
IF ALLOCATION (RECORD) THEN DO;
    DO I=1 TO NITEM;
        PUT FILE (OBJECT) EDIT (RECORD(I),'=Q_#',RECORD(I),' ');
        (A(12),A(4),A(12),A(1));
    END;
END;
IF LISTFLG=-1 THEN DO;
    PUT FILE (OBJECT) EDIT (' PUT FILE (',(FILENAM(I) DO I=1 TO
        FILECNT)) (A(11),32 A(1));
    PUT FILE (OBJECT) EDIT ('') EDIT (' RECORD IN ERROR') ('',
        'SKIP(2),A(16));') (A(29),A(15));
END;
PUT FILE (OBJECT) EDIT (' GO TO Q_#COMPLT; END;') (A(22));

/* DEAL WITH QUERY MESSAGE */
PUT FILE (OBJECT) EDIT (' IF Q_#QUERY THEN DO;',(FLGWORD(I)
    DO I=1 TO FLGCNT)) (A(21),32 A(1));
PUT FILE (OBJECT) EDIT (' =2;') (A(4));
IF LISTFLG=-1 THEN DO;
    PUT FILE (OBJECT) EDIT (' PUT FILE (',(FILENAM(I) DO I=1 TO
        FILECNT)) (A(11),32 A(1));
    PUT FILE (OBJECT) EDIT ('') EDIT (' RECORD QUERIED') ('',
        'SKIP(2),A(15));') (A(28),A(15));
END;
PUT FILE (OBJECT) EDIT (' GO TO Q_#COMPLT; END;') (A(22));

/* DEAL WITH ADJUST MESSAGE */
PUT FILE (OBJECT) EDIT (' IF Q_#ADJUST THEN DO;',(FLGWORD(I)
    DO I=1 TO FLGCNT)) (A(22),32 A(1));
```

```

1      ELSE DO;
2          LASTIF=0; LASTCOL=MATRIXCT;
3          RETURN; END;
4
5      /* COLUMN NOT THE SAME AS LAST. MUST BE TWO MORE IF LAST WAS IF */
6      NOTSAME: IF MATRIXCT>LASTCOL THEN DO;
7          IF LASTCOL=0 THEN GO TO ISTHISIF;
8          IF (MATRIXCT-LASTCOL)-1 THEN GO TO ILLEGAL;
9          IF LASTIF=0 THEN GO TO ILLEGAL;
10         GO TO ISTHISIF;
11         END;
12
13     /* COLUMN LESS THAN LAST. MUST BE IF ACTIVE IN PRESENT COLUMN */
14     IF IFFLG(MATRIXCT)=0 THEN GO TO BADIFBLK;
15     IF IFFLG(MATRIXCT)=-1 THEN GO TO ELSEPROC;
16     PUT FILE(OBJECT) EDIT (' GO TO ', LABLS(MATRIXCT), ': ');
17     (A(7),A(6),A(2)); /* WRITE GO TO TO SKIP AROUND */
18     LABSET(MATRIXCT)=1; /* IF-BLOCK ABOUT TO BE READ */
19     IF (FLAG=27)|(FLAG=28) THEN DO; /* THIS IS PART OF IF */
20     FLAG=28; END;
21     ELSE DO;
22     IF FLAG=5 THEN DO; /* THIS IS ELSE */
23     IFFLG(MATRIXCT)=-1; END;
24     ELSE GO TO NOELSE;
25     END;
26     LASTIF=1; LASTCOL=MATRIXCT; RETURN;
27
28     /* NO ELSE STATEMENT TO IF-GROUP IN PRESENT COLUMN */
29     NOELSE: PUT FILE (SYSPRINT) EDIT (' *** NO ELSE STATEMENT TO IF-',
30     'GROUP CLOSED BY THIS STATEMENT') (SKIP,A(29),A(30));
31     ERROR=1;
32     DO I=MATRIXCT TO 10; /*CLEAR ALL IF-FLAGS TO TRY AND SAVE */
33     IFFLG(I)=0; /*EXTRA ERRORS CAUSED BY THIS ONE */
34     LABSET(I)=0;
35     END;
36     LASTCOL=MATRIXCT; RETURN;
37
38     /* THIS STATEMENT IS FIRST AFTER LEGAL IF/ELSE GROUP */
39     ELSEPROC: DO I=MATRIXCT TO 10; /*ALL HIGHER IF-BLOCKS MUST HAVE ELSE */
40     IF IFFLG(I)=1 THEN GO TO NOELSE;
41     IF LABSET(I)=1 THEN DO; /* LABEL NEEDED, WRITE IT */
42     PUT FILE (OBJECT) EDIT (LABLS(I), ': ') (X(1),A(6),A(2));
43     LABLCT(I)=LABLCT(I)+1; /* UPDATE LABEL STORE */
44     PUT STRING (LABLS(I)) EDIT ('Q',I,'#',LABLCT(I))
45     (A(1),F(1,0),A(1),F(3,0));
46     LABSET(I)=0; IFFLG(I)=0;
47     END;
48     END;
49     GO TO ISTHISIF;
50     END COLCHK;

```



```

1 COLCHK: PROCEDURE;
2 /* FRED PRE-PROCESSOR DAVID MENDUS 4/3/67
3 PROCEDURE TO CHECK THE STARTING COLUMN OF EACH STATEMENT
4 FOR VALIDITY AND CONTROL THE IF-BLOCK STRUCTURE */
5
6 DECLARE (COLONE FIXED BINARY,
7 FLAG FIXED BINARY,
8 MATRXCT FIXED BINARY,
9 ERROR FIXED BINARY(1) ) STATIC EXTERNAL,
10
11 (LASTCOL FIXED BINARY,
12 IFFLG(10) FIXED BINARY(1),
13 LASTIF FIXED BINARY(1),
14 LABLS(10) CHARACTER(6),
15 LABLCT(10) FIXED BINARY,
16 CONT CHARACTER(1),
17 LABSET(10) FIXED BINARY(1),
18 FIRSTTIME FIXED BINARY(1) INITIAL(0) ) STATIC;
19
20 /* INITIALISE LABEL ARRAYS AND FLAGS FIRST TIME ONLY. */
21 IF FIRSTTIME THEN GO TO CHECKCOL;
22 LASTCOL, LASTIF, LABSET(1), LABSET(2), LABSET(3), LABSET(4),
23 LABSET(5), LABSET(6), LABSET(7), LABSET(8), LABSET(9),
24 LABSET(10), IFFLG(1), IFFLG(2), IFFLG(3), IFFLG(4), IFFLG(5),
25 IFFLG(6), IFFLG(7), IFFLG(8), IFFLG(9), IFFLG(10)=0;
26 LABLCT(1), LABLCT(2), LABLCT(3), LABLCT(4), LABLCT(5), LABLCT(6),
27 LABLCT(7), LABLCT(8), LABLCT(9), LABLCT(10)=1001;
28 LABLS(1)='Q1#001';
29 LABLS(2)='Q2#001';
30 LABLS(3)='Q3#001';
31 LABLS(4)='Q4#001';
32 LABLS(5)='Q5#001';
33 LABLS(6)='Q6#001';
34 LABLS(7)='Q7#001';
35 LABLS(8)='Q8#001';
36 LABLS(9)='Q9#001';
37 LABLS(10)='Q0#001';
38 FIRSTTIME=1;
39
40 /*CHECK FOR LEGAL COLUMN AND CONVERT TO 'MATRIX FORM' */
41 CHECKCOL: MATRXCT=1; DO I=7 TO 25 BY 2;
42 IF COLONE=I THEN GO TO LEGAL;
43 MATRXCT=MATRXCT+1;
44 END;
45 ILLEGAL: PUT FILE (SYSPRINT) EDIT (' *** STARTING COLUMN IS ILLEGAL')
46 (SKIP, A(31));
47 ERROR=1; LASTCOL=0; LASTIF=0; RETURN;
48
49 /* CHECK COLUMN WITH COLUMN OF LAST STATEMENT */
50 LEGAL: IF MATRXCT=LASTCOL THEN GO TO NOTSAME;
51
52 /* SAME COLUMN AS LAST TIME. O.K. UNLESS LAST STATEMENT WAS IF */
53 IFBLKCHK: IF IFFLG(MATRXCT)=0 THEN DO;
54 BADIFBLK: PUT FILE (SYSPRINT) EDIT (' *** IF-BLOCK STRUCTURE IS ',
55 'ILLEGAL') (SKIP, A(27), A(7));
56 ERROR=1; LASTCOL=0; LASTIF=0; RETURN;
57 END;
58 ISTHISIF: IF FLAG=1 THEN DO; /*IS THIS FIRST IF OF GROUP? */
59 IFFLG(MATRXCT)=1; LASTIF=1; LASTCOL=MATRXCT;
60 RETURN; END;

```



```

1
2 BRACKET: PROCEDURE;
3   /* FRED PRE-PROCESSOR, DAVID MENDUS, 9/3/67
4   TO READ AND WRITE A PARENTHESED EXPRESSION. USES ENTITY*/
5
6 DECLARE (ENDFLAG  FIXED BINARY(1),
7         SYNTAX(32) PACKED CHARACTER(1),
8         INTEND    FIXED BINARY(1),
9         SYNCNT    FIXED BINARY,
10        ERROR     FIXED BINARY(1)    ) STATIC EXTERNAL,
11        PARACT    FIXED BINARY STATIC;
12
13 CALLER: PARACT=1;                                /* INITIALISE COUNT */
14          CALL ENTITY;
15          IF ENDFLAG THEN DO;                      /* END OF STATEMENT TOO SOON */
16            PUT FILE (SYSPRINT) EDIT (' *** UNMATCHED PARENTHESES')
17            (SKIP,A(26)); INTEND=1;
18            ERROR=1; RETURN; END;
19          PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
20            (X(1),32 A(1));
21          IF SYNCNT=1 THEN GO TO CALLER;
22          IF SYNTAX(1)=')' THEN DO;                /* SEARCH FOR ) OR ( */
23            PARACT=PARACT-1;
24            IF PARACT=0 THEN RETURN;
25            GO TO CALLER;
26          END;
27          IF SYNTAX(1)='(' THEN PARACT=PARACT+1;
28          GO TO CALLER;
29        END BRACKET;
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

```

1      AIST:  PROCEDURE;
2              /* FRED PRE-PROCESSOR, DAVID MENDUS, 18/3/67
3              TO EDIT AND WRITE AN ASSIGNMENT STATEMENT */
4
5      DECLARE (ENDFLAG  BINARY FIXED (1),
6              SYNTAX(32) PACKED CHARACTER (1),
7              SYNCNT  BINARY FIXED,
8              ERROR    BINARY FIXED (1)      ) STATIC EXTERNAL;
9
10     PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
11     (X(1),32 A(1)); /* WRITE FIRST WORD OF LHS */
12     LHS:  CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
13           IF SYNTAX(1)='=' THEN GO TO RHS; /* IF LHS IS NOT ONE WORD */
14           IF SYNTAX(1)='(' THEN DO; /* THEN IT MUST BE ARRYA ELEMENT */
15             PUT FILE (OBJECT) EDIT ('(') (A(2));
16             CALL BRACKET; GO TO LHS; END;
17           ELSE DO;
18             PUT FILE (SYSPRINT) EDIT (' *** LHS OF ASSIGNMENT ILLEGAL')
19             (SKIP,A(30)); ERROR=1;
20           BADEND:  CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
21           END;
22
23     RHS:  PUT FILE (OBJECT) EDIT (' =') (A(2));
24           CALL ENTITY; IF ENDFLAG=0 THEN GO TO DORHS;
25     TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** ASSIGNMENT STATEMENT ',
26           'INCOMPLETE') (SKIP,A(26),A(10));
27           ERROR=1; RETURN;
28
29     DORHS: PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
30           (X(1),32 A(1)); /* ANYTHING GOES FOR RHS */
31     READRHS: CALL ENTITY;
32           IF SYNTAX(1)='(' THEN DO;
33             PUT FILE (OBJECT) EDIT ('(') (A(2));
34             CALL BRACKET; GO TO READRHS; END;
35           ELSE IF ENDFLAG=0 THEN GO TO DORHS;
36           PUT FILE (OBJECT) EDIT (';') (A(1)); RETURN;
37
38     END AIST;

```



```

1
2
3 ADST: PROCEDURE;
4 /* FRED PRE-PROCESSOR, DAVID MENDUS, 19/2/67
5 TO BEGIN PROCESSING AN ADJUST STATEMENT. ADCON DOES THE
6 DIAGNOSTIC PROCESSING. */
7
8 DECLARE (ENDFLAG FIXED BINARY (1),
9 SYNTAX(32) PACKED CHARACTER (1),
10 SYNCNT FIXED BINARY,
11 ERROR FIXED BINARY (1),
12 NITEM FIXED BINARY ) STATIC EXTERNAL,
13 RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL,
14 WORD CHARACTER(32) DEFINED SYNTAX;
15
16 /* CHECK THAT THE VARIABLE TO BE ADJUSTED IS VALID */
17 CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
18 IF ALLOCATION(RECORD) THEN DO;
19 DO I=1 TO NITEM;
20 IF WORD=RECORD(I) THEN GO TO WRITE;
21 END; /* ERROR IF NO MATCH */
22 PUT FILE (SYSPRINT) EDIT (' *** ADJUSTMENT IS TO INVALID ',
23 'ITEM') (SKIP,A(30),A(4));
24 ERROR=1; GO TO BADEND;
25 END;
26 WRITE: PUT FILE (OBJECT) EDIT ('Q #ADJUST=Q #ADJUST+1; ',
27 (SYNTAX(I) DO I=1 TO SYNCNT)) (A(23),32 A(1));
28 WHERESEQ: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
29 IF SYNTAX(1)='(' THEN DO; /* PARENTHESES LEGAL */
30 PUT FILE (OBJECT) EDIT (' (' (A(2));
31 CALL BRACKET; GO TO WHERESEQ;
32 END;
33 IF SYNTAX(1)!='=' THEN DO; /* ELSE MUST BE EQUALS */
34 PUT FILE (SYSPRINT) EDIT (' *** ASSIGNMENT FORMAT INVALID')
35 (SKIP,A(30));
36 ERROR=1; GO TO BADEND;
37 END;
38 PUT FILE (OBJECT) EDIT (' =') (A(2)); /* FOUND */
39 CALL ENTITY;
40 RHS: IF ENDFLAG THEN DO; /* STATEMENT INCOMPLETE */
41 TOOSOON: PUT FILE (SYSPRINT) EDIT (' *** ADJUST STATEMENT IS ',
42 'INCOMPLETE') (SKIP,A(25),A(10));
43 ERROR=1; RETURN;
44 END;
45 IF SYNTAX(1)='(' THEN DO;
46 PUT FILE (OBJECT) EDIT (' (' (A(2));
47 CALL BRACKET; GO TO RHS;
48 END;
49 IF SYNTAX(1)=',' THEN DO; /* ENDS THIS SECTION */
50 PUT FILE (OBJECT) EDIT (' ;') (A(2));
51 CALL ADCON; RETURN;
52 END;
53 PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
54 (X(1),32 A(1));
55 GO TO RHS;
56 BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
57 END ADST;

```


ATTRIBUTE TABLE

DCL NO.	IDENTIFIER	ATTRIBUTES
	ABS	GENERIC,BUILT-IN FUNCTION
2	BILLED	AUTOMATIC,DECIMAL,FIXED(7,2)
1	BILLER	ENTRY,DECIMAL,FLOAT(SINGLE)
2	CARD	AUTOMATIC,STRING,CHARACTER
2	COST	AUTOMATIC,DECIMAL,FIXED(7,2)
2	CUSTOMER	AUTOMATIC,DECIMAL,FIXED(5,0)
2	DISCOUNT	AUTOMATIC,DECIMAL,FIXED(7,2)
2	DISCRATE	AUTOMATIC,DECIMAL,FIXED(5,0)
10	ENDING	STATEMENT LABEL CONSTANT
2	ERRCODE	AUTOMATIC,BINARY,FIXED(15,0)
2	FREIGHT	AUTOMATIC,DECIMAL,FIXED(7,2)
2	FRTRATE	AUTOMATIC,DECIMAL,FIXED(5,2)
2	INSRATE	AUTOMATIC,DECIMAL,FIXED(5,0)
2	INSURANCE	AUTOMATIC,DECIMAL,FIXED(7,2)
11	INVOICE	ENTRY,BINARY,FIXED(15,0)
	IQ_#	AUTOMATIC,BINARY,FIXED(15,0)
2	ITEM	AUTOMATIC,DECIMAL,FIXED(5,0)
37	Q1#001	STATEMENT LABEL CONSTANT
45	Q1#002	STATEMENT LABEL CONSTANT
71	Q1#003	STATEMENT LABEL CONSTANT
79	Q1#004	STATEMENT LABEL CONSTANT
95	Q1#005	STATEMENT LABEL CONSTANT
111	Q1#006	STATEMENT LABEL CONSTANT
123	Q1#007	STATEMENT LABEL CONSTANT
129	Q1#008	STATEMENT LABEL CONSTANT
71	Q2#001	STATEMENT LABEL CONSTANT

DCL NO.	IDENTIFIER	ATTRIBUTES
95	Q2#002	STATEMENT LABEL CONSTANT
111	Q2#003	STATEMENT LABEL CONSTANT
12	Q_#ADJUST	AUTOMATIC,BINARY,FIXED(15,0)
13	Q_#BILLED	AUTOMATIC,DECIMAL,FIXED(7,2)
162	Q_#COMPLT	STATEMENT LABEL CONSTANT
13	Q_#COST	AUTOMATIC,DECIMAL,FIXED(7,2)
13	Q_#CUSTOMER	AUTOMATIC,DECIMAL,FIXED(5,0)
12	Q_#DIAG	(*),AUTOMATIC,PACKED,STRING,CHARACTER
12	Q_#DIGCT	AUTOMATIC,BINARY,FIXED(15,0)
13	Q_#DISCOUNT	AUTOMATIC,DECIMAL,FIXED(7,2)
13	Q_#DISCRATE	AUTOMATIC,DECIMAL,FIXED(5,0)
12	Q_#FATAL	AUTOMATIC,BINARY,FIXED(1,0)
13	Q_#FREIGHT	AUTOMATIC,DECIMAL,FIXED(7,2)
13	Q_#FRTRATE	AUTOMATIC,DECIMAL,FIXED(5,2)
13	Q_#INSRATE	AUTOMATIC,DECIMAL,FIXED(5,0)
13	Q_#INSURANCE	AUTOMATIC,DECIMAL,FIXED(7,2)
13	Q_#ITEM	AUTOMATIC,DECIMAL,FIXED(5,0)
12	Q_#QUERY	AUTOMATIC,BINARY,FIXED(1,0)
13	Q_#UNITQTY	AUTOMATIC,DECIMAL,FIXED(5,0)
13	Q_#UNITVAL	AUTOMATIC,DECIMAL,FIXED(7,2)
6	READ	STATEMENT LABEL CONSTANT
	SYSIN	FILE,EXTERNAL
	SYSPRINT	FILE,EXTERNAL
2	UNITQTY	AUTOMATIC,DECIMAL,FIXED(5,0)
2	UNITVAL	AUTOMATIC,DECIMAL,FIXED(7,2)

BILLER: PROCEDURE OPTIONS(MAIN);

PAGE

6

COMPILER DIAGNOSTIC MESSAGES

THE SUPPRESSION OF WARNING MESSAGES HAS BEEN REQUESTED.

TIME FOR THIS COMPILATION WAS 1.14 MINUTES

123877

*** INVOICE EDIT RUN ***

RECORD IN ERROR

1
0
0.00

137
0.00
15.65

12
0

1.00 1200.00
0.00 3.65
BILLED COST DOES NOT AGREE
COST INVALID FOR QUANTITY

RECORD ADJUSTED

21.70 20
1.34 0.06

219
4.34
18.70

10
0

2.17
0.00

DISCOUNT RECALCULATED

RECORD IN ERROR

342.60 10
0.00 0.00

486
34.26
307.34

10
0

34.26
0.00

BILLED COST DOES NOT AGREE
DISCOUNT RATE INVALID

RECORD IN ERROR

180.00 4
1.40 0.00

4861
0.00
185.00

100
2

1.80
3.60

ITEM CODE OUT OF RANGE

RECORD QUERIED

60.00 5
10.00 0
0.16

327
0.00
73.00

1000
5

0.06
3.00

FREIGHT HIGH

RECORD IN ERROR

142.00 6
3.61 0
0.02

998
0.00
139.61

100
0

1.42
0.00

BILLED COST DOES NOT AGREE

RECORD IN ERROR

27.00 99997
1.96 0
0.07

376
0.00
30.31

100
5

0.27
1.35

CUSTOMER CODE OUT OF RANGE

RECORD IN ERROR

13.90 8
16.32 7
1.17

1031
2.46
514.60

10
2

1.39
3.19

BILLED COST DOES NOT AGREE
DISCOUNT RATE INVALID

INSURANCE MISCALCULATED
FREIGHT HIGH

RECORD IN ERROR

24.60 9
1.00 20
0.04

2476
4.92
30.52

10
0

2.46
0.00

BILLED COST DOES NOT AGREE

RECORD ADJUSTED

1605.40 10
0.00 0
0.00

2581
0.00
1637.50

20
2

80.27
32.10

INSURANCE RECALCULATED

123882

```
112      IGH = 0) THEN DO; GO TO Q1#007; END; IF (COST / FREIGHT < 10) THEN DO; Q_#QUER
118      Y=1; IF Q_#DIGCT=40 THEN Q_#DIGCT=Q_#DIGCT+1; Q_#DIAG(Q_#DIGCT)='FREIGHT HIGH';
121      GO TO Q1#007; END; Q1#007: IF (COST = 0) THEN DO; FRTRATE, Q_#FRTRATE= 0; GO
127      TO Q1#008; END; FRTRATE, Q_#FRTRATE= FREIGHT / COST; Q1#008: IF Q_#ADJUST>2 T
130      HEN Q_#FATAL=1; IF Q_#FATAL THEN DO; ERRCODE=3; ITEM =Q_#ITEM; CUS
136      TOMER =Q_#CUSTOMER; UNITQTY =Q_#UNITQTY; DISCRATE =Q_#DISCRATE
138      ;INSRATE =Q_#INSRATE; UNITVAL =Q_#UNITVAL; DISCOUNT =Q_#DI
141      SCOUNT; INSURANCE =Q_#INSURANCE; FREIGHT =Q_#FREIGHT; COST
144      =Q_#COST; BILLED =Q_#BILLED; FRTRATE =Q_#FRTRATE; PUT
147      FILE (SYSPRINT) EDIT (' RECORD IN ERROR') (SKIP(2),A(16)); GO TO Q_#COMPLT; END
149      ; IF Q_#QUERY THEN DO; ERRCODE=2; PUT FILE (SYSPRINT) EDIT (' RECORD QUERIED') (
153      SKIP(2),A(15)); GO TO Q_#COMPLT; END; IF Q_#ADJUST THEN DO; ERRCODE=1; PUT FILE
159      (SYSPRINT) EDIT (' RECORD ADJUSTED') (SKIP(2),A(16)); GO TO Q_#COMPLT; END; ERRCO
162      DE=0; Q_#COMPLT; IF (Q_#FATAL)|(Q_#QUERY)|(Q_#ADJUST) THEN DO; PUT FILE (SYSPRINT)
165      LIST (CUSTOMER, ITEM, UNITQTY, UNITVAL, COST, DISCRATE, DISCOUNT, INSRATE, INSURANCE, FR
169      EIGHT, FRTRATE, BILLED); END; IF Q_#DIGCT=0 THEN DO; PUT FILE (SYSPRINT) EDIT ((Q
      _#DIAG(IQ_#) DO IQ_#=1 TO Q_#DIGCT)) (SKIP,A(40)); END; END INVOICE; END BILLER;
```

END OF LISTING ***