RA

stichting
mathematisch
centrum

MC

REKENAFDELING RA NR 17/71 JUNI

D. GRUNE
BESCHRIJVING VAN DE TEKSTCORRECTOR
DE PROCEDURES 'ALTER' EN 'TC PUNCH'

2e boerhaavestraat 49 amsterdam

1. Doel.

De procedures alter en to punch zijn de hoofdbouwstenen van het programma tcpp, tekstcorrector van papierband naar papierband.

1.1. De procedure alter heeft tot doel, uitgaande van een symbolenstroom met correctie-instructies en een symbolenstroom met informatie, een nieuwe symbolenstroom te produceren volgens bepaalde specificaties. Alle symboolstromen worden beschreven door parameterevaluaties; de procedure alter voert zelf geen enkele in- of uitvoer-opdracht uit.

1.2. De procedure to punch verzorgt de ponsbanduitvoer volgens een aantal bijzondere eisen.

2. De procedure alter.

De handleiding bij het programma tcpp (LR 2.2) wordt bekend verondersteld.

2.1. De procedure heeft de volgende heading:

```
integer procedure alter(corsym,datsym,result,repsym,available);
value available;
integer corsym,datsym,available;
procedure result,repsym;
```

2.2. De parameters hebben de volgende betekenis:

2.2.1. integer corsym (by name)
Opeenvolgende evaluaties van corsym leveren de interne waarden van opeenvolgende symbolen van de correctie-instructies af. Deze waarden mogen niet groter dan 127 zijn; grotere waarden worden als onbekende symbolen behandeld.

Bijzondere waarden:

waarde < 0 ∧ waarde ≠ -4096: onbekend symbool
waarde = -4096: einde invoerstroom

Als actuele parameter kan b.v. resymbol optreden.

2.2.2. integer datsym (by name)
Opeenvolgende evaluaties van datsym leveren de interne waarden af van opeenvolgende symbolen van de informatiestroom waarop de correcties aangebracht moeten worden. Deze waarden mogen niet groter zijn dan 127; grotere waarden worden als onbekende symbolen behandeld.

Bijzondere waarden:

waarde < 0 ∧ waarde ≠ -4096: onbekend symbool
waarde = -4096: einde invoerstroom

Na de eerste evaluatie van de parameter datsym wordt de parameter corsym niet meer geevalueerd; beide invoerstromen kunnen dus achter elkaar via resymbol ingelezen worden.

2.2.3. procedure result(n); integer n;
De interne representaties van opeenvolgende gegenereerde symbolen worden als parameter meegegeven aan opeenvolgende aanroepen van de procedure result. De hierdoor verkregen uitvoerstroom komt overeen met de resultband uit de beschrijving van tcpp.

Bijzondere waarden van de parameter n:

-1: onbekend symbool,

en verder alle waarden die via de instructie 'td' doorgegeven worden; deze zijn positief en kleiner dan 2 ↑ 26.

2.2.4. procedure repsym(n); integer n;
De interne representaties van opeenvolgende af te drukken symbolen worden als parameter meegegeven aan opeenvolgende aanroepen van de procedure repsym. De hierdoor verkregen informatiestroom bevat het verslag van de correctiehandelingen.

Bijzondere waarde van de parameter n:

113: new page, wordt alleen gebruikt als scheider tussen de afdruk van de correctie-instructies en de afdruk van de resultstroom.

2.2.5. integer available (by name)
De procedure alter gebruikt de parameter available om de grootte te bepalen van het werkarray voor de correctie-instructies. De procedure alter kan de standaardfunctie available niet gebruiken, omdat volgende evaluaties van corsym en datsym of aanroepen van result of repsym onbepaalde hoeveelheden geheugen kunnen opeisen.

2.3. De door alter afgeleverde integer waarde heeft de volgende betekenis:

0. foutloos verloop, nieuwe informatiestroom geproduceerd volgens specificaties.

1. fout in de correctie-instructies, geen nieuwe informatiestroom geproduceerd.

2. fout tijdens corrigeren, nieuwe informatiestroom geproduceerd, maar niet geheel volgens specificaties.

2.4. Beperkingen.

2.4.1. De procedure is niet beveiligd tegen overflow in het werkarray. In de huidige versie van tcpp is hierin plaats voor ongeveer 90000 symbolen in 'ib'-instructies, of ongeveer 30000 symbolen in de andere instructies.

2.4.2. Regelnummers groter dan 99999 worden foutief afgedrukt.

2.4.3. Doorgegeven waarden bij de instructie 'td' welke groter zijn dan 99999 worden foutief afgedrukt; ze worden wel correct doorgegeven.

3. De procedure to punch.

3.1. De procedure heeft de volgende heading:

Boolean procedure to punch(symbol); value symbol;
integer symbol;

3.2. De procedure to punch voert het symbool waarvan de interne representatie gelijk is aan 'symbol' over de bandponser uit, waarbij de volgende regels in acht genomen worden:

1. Tabs en spaties worden op plaatsen waar ze niet in de zichtbare tekst tot uiting komen, niet geponst. Bij voorbeeld worden spaties aan het eind van een regel niet geponst.

2. Tabs worden alleen aan het begin van een regel gebruikt, en dan ook optimaal. Bij voorbeeld worden 9 spaties gevolgd door een letter aan het begin van een regel geponst als tab en spatie + letter.

3. Elk symbool wordt hoogstens eenmaal onderstreept en/of doorbalkt.

4. Een onderstreping en/of doorbalking aan het eind van een regel wordt van een drager (spatie) voorzien.

5. Een symboolwaarde die geen interne representatie van een ponsband symbool is, geeft aanleiding tot het ponsen van een markering, een aantal schuine banen voorafgegaan en gevolgd door runout. (zie echter 3.3.).

6. Op regelmatige afstanden wordt automatisch runout gegeven, bij voorkeur bij een blanke regel; dit gebeurt nooit in het midden van een regel. De ponsingen na runout bevatten een caseponsing vooafgaand aan het eerste casegevoelige symbool.

7. Bij een opeenvolging van meerdere blanke regels wordt de automatische runout gegeven voorafgaand aan het laatste nlcr-symbool.

3.3. Naast de gewone pusymrepresentaties kent to punch de volgende waarden:

110: punch-off, het ponsen van symbolen wordt tijdelijk stopgezet.

111: punch-on, het ponsen van symbolen wordt hervat.

113: new tape, een bandscheider bestaande uit stopcode, runout, erase, runout, runout.

114: runout.

3.4. De door to punch afgeleverde Boolean waarde is false indien een markering zoals genoemd in 3.2.5. geponst moest worden, en true in alle andere gevallen.

3.5. Gebruik.

Voor de procedure to punch moeten de volgende globalen gedeclareerd worden:

```
integer inpos, outpos, np;
Boolean is punch, is undl, is vertb, wlle;
integer array punchlist[0:127];
procedure init punchlist;
```

De variabelen worden geinitialiseerd door een aanroep van init punchlist; deze procedure behoeft maar eenmaal per programma aangeroepen te worden.

3.6. Code.

De procedure is gericht op flexowriter-code, maar kan door een paar eenvoudige wijzigingen geschikt gemaakt worden voor ISO-code. Het array punchlist moet dan uitgebreid worden en de erase moet geponst worden door puhep(255).

4. Bijlage.

Bijgevoegd is een afdruk van het programma tcpp, waarin zowel alter als to punch voorkomen.

B2 73V.250,DICK GRUNE

```
BEGIN   COMMENT TEXT CORRECTOR, VERSION: 14 - 5 - 1971;

             INTEGER INPOS, OUTPOS, NP;   BOOLEAN IS PUNCH, IS UNDL, IS VERTB, WLLE;
             INTEGER ARRAY PUNCHLIST[0:135];

             PROCEDURE INIT PUNCHLIST;
             BEGIN   INTEGER I;
                     INPOS:= OUTPOS:= NP:= 0;
                     IS PUNCH:= TRUE; IS UNDL:= IS VERTB:= WLLE:= FALSE;
                     FOR I:= 0 STEP 1 UNTIL 135 DO
                     PUNCHLIST[I]:= STRINGSYMBOL(I,
                     ¢NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNINNNNNNNNNNNNNNNNNNNNNNNNNNINNNNNNNNNNNNNNNNNIIIIIINNNNNISIIIINNNNNNIIIIIISSISSIIIS
SNNNIIINNIII NNNN≯);
                     IF PUNCHLIST[135] = 255 THEN BEGIN PRINTTEXT(¢***ERROR IN INIT PUNCHLIST≯);   EXIT   END
             END INIT PUNCHLIST;

             BOOLEAN PROCEDURE TO PUNCH(SYM);   VALUE SYM;   INTEGER SYM;
             BEGIN   INTEGER TYPE;

                     PROCEDURE MARKER;
                     BEGIN   INTEGER CNT,S;
                             RUNOUT;    S:= 67;
                             FOR CNT:= 1 STEP 1 UNTIL 126 DO
                             BEGIN   PUHEP(S);
                                     S:= S + S;    IF S >127 THEN S:= S - 127
                             END CNT;
                             RUNOUT;
                             TO PUNCH:= FALSE
                     END MARKER;

                     TO PUNCH:= TRUE;
                     IF ¬ IS PUNCH THEN IS PUNCH:= SYM = 111 ELSE
                     IF SYM < 0  ∨  SYM > 135 THEN MARKER ELSE
                     BEGIN   COMMENT LEGAL SYMBOL;
                             TYPE:= PUNCHLIST[SYM];
                             IF TYPE  = 23
                             THEN    BEGIN   COMMENT PRINTABLE SYMBOL;
                                             IF INPOS ≠ OUTPOS
                                             THEN    BEGIN   COMMENT OUTPUT LAGS BEHIND;
                                                             IF OUTPOS ≤ 0
                                                             THEN    BEGIN   COMMENT NO OUTPUT YET ON THIS LINE;
                                                                             IF OUTPOS = -1
                                                                             THEN    BEGIN   COMMENT NLCR NOT YET PUNCHED;
                                                                                             IF NP > 400  ∨  (WLLE  ∧  NP > 100)
                                                                                             THEN    BEGIN   RUNOUT;    NP:= 0
                                                                                                     END
                                                                                             FI;

                                                                                             PUNLCR;    NP:= NP + 1;    OUTPOS:= 0
                                                                                     END
                                                                             FI;

                                                                     TAB:    IF INPOS ≥ OUTPOS + 8
                                                                             THEN    BEGIN   PUSYM(118);    OUTPOS:= OUTPOS + 8;
                                                                                             NP:= NP + 1;    GOTO TAB
                                                                                     END
```

```
56                                                          FI
57                                                  END TABS
58                                          FI;
59
60                                          PUSPACE(INPOS - OUTPOS);   NP:= NP + INPOS - OUTPOS;   OUTPOS:= INPOS;
61                                          IS UNDL:= IS VERTB:= FALSE
62                                  END BLANKS
63                          FI;
64
65                          IF ~(IS UNDL ^ SYM = 126  v  IS VERTB ^ SYM = 127)
66                          THEN    BEGIN   COMMENT AVOID DUPLICATING UNDERLINES OR BARS;
67
68                                  IF SYM < 126
69                                  THEN    BEGIN   OUTPOS:= OUTPOS + 1;   INPOS:= INPOS + 1;
70                                                  IS UNDL:= IS VERTB:= FALSE
71                                          END NOT UNDERLINE, NORMAL SYMBOL
72                                  ELSE    BEGIN   IF SYM = 126 THEN IS UNDL:= TRUE ELSE IS VERTB:= TRUE END
73                                          FI;
74
75                                  NP:= NP + 1;   PUSYM(SYM)
76                                  END
77                          FI
78                  END PRINTABLE SYMBOL
79          ELSE
80                  IF TYPE = 18
81                  THEN MARKER
82                  ELSE    IF SYM < 113
83                          THEN    BEGIN   IF SYM < 110
84                                          THEN INPOS:= INPOS + 1
85                                          ELSE IS PUNCH:= SYM = 111
86                                  END SPACE, PUNCH-OFF, PUNCH-ON
87                          ELSE    IF SYM < 118
88                                  THEN    BEGIN   COMMENT RUNOUTS;
89                                          IF OUTPOS = -1 THEN BEGIN PUNLCR;   OUTPOS:= 0 END;
90                                          IF SYM = 113
91                                          THEN    BEGIN   PUHEP(0);   PUHEP(0);   PUHEP(0);
92                                                          STOPCODE;   RUNOUT;
93                                                          PUHEP(127);   RUNOUT
94                                                  END TAPE SEPARATOR
95                                          FI;
96
97                                          RUNOUT:   NP:= 0
98                                  END RUNOUTS
99                                  ELSE    IF SYM < 119
00                                          THEN INPOS:= ((INPOS + 1) ÷ 8 + 1) * 8
01                                          ELSE    BEGIN   COMMENT NLCR;
02                                                  IF OUTPOS = -1
03                                                  THEN    BEGIN   PUNLCR;   NP:= NP + 1;
04                                                                  WLLE:= TRUE
05                                                          END EMPTY LINE
06                                                  ELSE    BEGIN  WLLE:= FALSE END
07                                                  FI;
08
09                                                  IF IS UNDL  v  IS VERTB
10                                                  THEN    BEGIN   PUSPACE(1);   NP:= NP + 1;
11                                                                  IS UNDL:= IS VERTB:= FALSE
12                                                          END
13                                                  FI;
14
15                                                  INPOS:=  0;   OUTPOS:= -1
```

```
                                                    END NLCR
                                                ELSE
                                            ELSE
                                        ELSE
                                    ELSE
                                ELSE
                        END LEGAL SYMBOL
    END TO PUNCH;

    PROCEDURE REPSYM(SYM);  VALUE SYM;  INTEGER SYM;
    COMMENT NEWPAGE ON LINE-PRINTER;
    IF SYM = 113 THEN NEW PAGE ELSE PRSYM(SYM);

    INTEGER NLCR COUNT;

    PROCEDURE SEP FIRST LINE(SYM);  VALUE SYM;  INTEGER SYM;
    COMMENT CAUSES RUNOUT AFTER THE FIRST LINE,
            FOR THE BENEFIT OF THE MONITOR HEADING;
    IF SYM = 119 THEN
    BEGIN NLCR COUNT:= NLCR COUNT + 1;  TO PUNCH(119);  IF NLCR COUNT = 2 THEN TO PUNCH(114)  END
    ELSE TO PUNCH(SYM);

    INTEGER PROCEDURE ALTER(CORSYM, DATSYM, RESULT, REPSYM, AVAILABLE);  VALUE AVAILABLE;
    INTEGER CORSYM, DATSYM, AVAILABLE;  PROCEDURE RESULT, REPSYM;
    BEGIN

            COMMENT GENERAL TOOLS:  ;

            INTEGER NLCR SYMBOL, UNDERLINE,VERTICAL BAR, SPACE SYMBOL, TAB SYMBOL, NEWPAGE SYMBOL,
                    CROSS, STAR, UNDEFINED, INTERN UNDEF, END, INTERN END, STOP,
                    LAST CHARACTER, CHARACTER IN STOCK, MEMORY POINTER;

            BOOLEAN ERRORS DETECTED, PASS 1;

            INTEGER ARRAY MEMORY[0 : AVAILABLE - 500];

            INTEGER PROCEDURE CHARIN;
            BEGIN   INTEGER UNDERLINE MODIFIER, BAR MODIFIER, N;
                    UNDERLINE MODIFIER:= BAR MODIFIER:= 0;
            NEXT:
                    IF CHARACTER IN STOCK ≥ 0 THEN BEGIN N:= CHARACTER IN STOCK;  CHARACTER IN STOCK:= -1 END ELSE
                    BEGIN   N:= IF PASS 1 THEN CORSYM ELSE DATSYM;
                            IF N > 127 THEN N:= -1
                    END GET LIMITED SYMBOL;

                    IF N < 0 THEN
                    BEGIN   IF N= END THEN
                            BEGIN   IF PASS 1 THEN
                                    BEGIN   ERROR(‡INPUT EXHAUSTED$);  GOTO EXIT  END ELSE
                                    BEGIN N:= INTERN END;  CHARACTER IN STOCK:= STOP END
                            END END ELSE

                            BEGIN   IF PASS 1 THEN
                                    BEGIN   ERROR(‡UNDEFINED SYMBOL ENCOUNTERED$);  ALTER:= 1;
                                            ERRORS DETECTED:= TRUE;  N:= SPACE SYMBOL
                                    END ELSE
                                    N:= INTERN UNDEF
                            END UNDEFINED
                    END SPECIAL ELSE
```

```
                IF N < NLCR SYMBOL THEN ELSE

                IF N= UNDERLINE THEN    BEGIN UNDERLINE MODIFIER:= 128; GOTO NEXT END ELSE
                IF N= VERTICAL BAR THEN BEGIN BAR MODIFIER:= 256; GOTO NEXT END ELSE

                IF N= NLCR SYMBOL THEN
                BEGIN   IF UNDERLINE MODIFIER + BAR MODIFIER ≠ 0 THEN
                        BEGIN  CHARACTER IN STOCK:= N;   N:= SPACE SYMBOL END
                END NLCR SYMBOL ELSE

                IF N = STOP THEN
                BEGIN   NEW LINE;
                        IF EXEC RE THEN GOTO EXIT;
                        REPSYM(NLCR SYMBOL);   REPSYM(NLCR SYMBOL);   ALTER:= 2;
                        IF DELETE THEN
                        ERROR(‡END OF FILE ENCOUNTERED WHILE DELETING‡) ELSE
                        ERROR(‡END OF FILE ENCOUNTERED WHILE COPYING‡);
                        GOTO EXIT
                END STOP;

                LAST CHARACTER:= CHARIN:= N + UNDERLINE MODIFIER + BAR MODIFIER;
                IF PASS 1 THEN CHAROUT(LAST CHARACTER) ELSE
                IF LAST CHARACTER = NLCR SYMBOL THEN INPUT LINE NUMBER:= INPUT LINE NUMBER + 1
        END CHARIN;

        PROCEDURE CHAROUT(CHARACTER);   VALUE CHARACTER;   INTEGER CHARACTER;
        BEGIN   IF CHARACTER>255 THEN
                BEGIN   REPSYM(VERTICAL BAR);   CHARACTER:= CHARACTER - 256 END;
                IF CHARACTER>127 THEN
                BEGIN   REPSYM(UNDERLINE);   CHARACTER:= CHARACTER - 128 END;
                IF CHARACTER = INTERN END   THEN ELSE
                IF CHARACTER = INTERN UNDEF THEN
                BEGIN   ERROR(‡UNDEFINED SYMBOL ENCOUNTERED‡);
                        ALTER:= 2;   REPSYM(SPACE SYMBOL)
                END INTERN UNDEF ELSE
                REPSYM(CHARACTER)
        END CHAROUT;

        BOOLEAN PROCEDURE LAYOUT(CHARACTER); VALUE CHARACTER; INTEGER CHARACTER;
        LAYOUT:= CHARACTER= SPACE SYMBOL ∨ CHARACTER= TAB SYMBOL;

        PROCEDURE ERROR(MESSAGE);   STRING MESSAGE;
        BEGIN   INTEGER SYMBOL COUNTER, SYMBOL, SAVE PRINT POS;
                SYMBOL COUNTER:= -2;   SAVE PRINT POS:= PRINT POS;
                FOR SYMBOL:= NLCR SYMBOL,   NLCR SYMBOL,
                        STRINGSYMBOL(SYMBOL COUNTER, MESSAGE) WHILE SYMBOL≠ 255,
                        SPACE SYMBOL WHILE SYMBOL COUNTER < 120,
                        STRINGSYMBOL(SYMBOL COUNTER - 120, ‡*** ERROR ***‡) WHILE SYMBOL ≠ 255,
                        NLCR SYMBOL,   NLCR SYMBOL,
                        SPACE SYMBOL WHILE PRINT POS ≠ SAVE PRINT POS              DO
                BEGIN   REPSYM(SYMBOL);   SYMBOL COUNTER:= SYMBOL COUNTER + 1   END
        END ERROR;

        COMMENT INPUT OF CORRECTIONS:  ;

        INTEGER DC, DL, DN, DX, DE, CC, CL, CN, CX, CE, IC, IL, SN, EC, IB, RE, TD, ST, CO;

        PROCEDURE READ OPERATION TAPE;
        BEGIN   INTEGER INSTRUCTION;   BOOLEAN IN ERROR RECOVERY;
                SWITCH PROCESS:=        PDC, PDL, PDN, PDX, PDE,
```

```
                                        PCC, PCL, PCN, PCX, PCE,
                                        PIC, PIL, PSN, PEC, PIB,
                                        PRE, PTD, PST, PCO, PER;

        NEXT:   IN ERROR RECOVERY:= FALSE;
        BAD:    INSTRUCTION:= READ INSTRUCTION;   STORE(INSTRUCTION);
                GOTO PROCESS[INSTRUCTION];

        PDC:    PDL:    PDX:    PCC:    PCL:    PCX:    PEC:
                READ STRING(TRUE);   GOTO NEXT;

        PDN:    PDE:    PCN:    PCE:    PSN:    PTD:
                STORE(1);   STORE(INTEGER);   SKIP LINE;   GOTO NEXT;

        PIC:    PIL:
                READ STRING(FALSE);   GOTO NEXT;

        PIB:    READ BLOCK;   SKIP LINE;   GOTO NEXT;

        PCO:    ERASE;   SKIP COMMENT;   SKIP LINE;   GOTO NEXT;

        PER:    SKIP LINE;
                IF ~IN ERROR RECOVERY THEN
                BEGIN   ERRORS DETECTED:= IN ERROR RECOVERY:= TRUE;   ERROR(‡UNKNOWN OPERATION‡);   ALTER:= 1   END;
                GOTO BAD;

        PRE:    PST:
                SKIP LINE

        END READ OPERATION TAPE;

        PROCEDURE ERASE;   MEMORY POINTER:= MEMORY POINTER - 1;

        INTEGER PROCEDURE INTEGER;
        BEGIN   INTEGER N;   REAL RESULT;
                RESULT:= 0;   N:= LAST CHARACTER;
        SKIP LAYOUT:
                IF LAYOUT(N) THEN BEGIN N:= CHARIN;   GOTO SKIP LAYOUT   END;
                IF N<10 THEN
                BEGIN   RESULT:= RESULT * 10 + N;
                        N:= CHARIN;
                        GOTO SKIP LAYOUT
                END ONE DIGIT;
                IF RESULT > MAXINT THEN
                BEGIN ERROR(‡INTEGER OVERFLOW‡);  RESULT:= 0 END;
                INTEGER:= RESULT
        END INTEGER;

        PROCEDURE SKIP COMMENT;
        BEGIN   INTEGER TERMINATOR, N;
                TERMINATOR:= LAST CHARACTER;
        NEXT: N:= CHARIN;
                IF N ≠ TERMINATOR THEN GOTO NEXT
        END SKIP COMMENT;

        INTEGER PROCEDURE READ INSTRUCTION;
        BEGIN   INTEGER N, OPERATION CODE, I, TEST;
                N:= LAST CHARACTER;
        SKIP LAYOUT:
                IF LAYOUT(N) ∨ N=NLCR SYMBOL  THEN BEGIN N:= CHARIN; GOTO SKIP LAYOUT END;
```

```
296                     := CHARIN;
297                     IF N > 36 THEN N:= N - 27;
298                     IF I > 36 THEN I:= I - 27;
299                     OPERATION CODE:= N * 512 + I; I:= 1;
300                     FOR TEST:= DC, DL, DN, DX, DE,
301                                CC, CL, CN, CX, CE,
302                                IC, IL, SN, EC, IB,
303                                RE, TD, ST, CO          DO
304                     IF TEST= OPERATION CODE THEN GOTO FOUND ELSE I:= I + 1;
305             FOUND:  READ INSTRUCTION:= I;
306                     IF LAYOUT(CHARIN) THEN CHARIN
307             END READ INSTR;
308
309             INTEGER PROCEDURE CONVERT INSTRUCTION(INSTRUCTION CODE);  STRING INSTRUCTION CODE;
310             CONVERT INSTRUCTION:= STRINGSYMBOL(0, INSTRUCTION CODE) * 512 + STRINGSYMBOL(1, INSTRUCTION CODE);
311
312             PROCEDURE READ STRING(INPUT IS TO BE CONDENSED); VALUE INPUT IS TO BE CONDENSED; BOOLEAN INPUT IS TO BE CONDENSED;
313             BEGIN   INTEGER N, BEGINNING OF STRING;
314                     N:= LAST CHARACTER; BEGINNING OF STRING:= MEMORY POINTER;
315                     STORE(0);
316             NEXT:   IF N ≠ NLCR SYMBOL THEN
317                     BEGIN   IF ¬ (LAYOUT(N) ∧ INPUT IS TO BE CONDENSED) THEN
318                             STORE(N);
319                             N:= CHARIN; GOTO NEXT
320                     END;
321                     MEMORY[BEGINNING OF STRING]:= MEMORY POINTER - BEGINNING OF STRING - 1
322             END READ STRING;
323
324             PROCEDURE STORE(CHARACTER);  VALUE CHARACTER;  INTEGER CHARACTER;
325             BEGIN   MEMORY[MEMORY POINTER]:= CHARACTER;
326                     MEMORY POINTER:= MEMORY POINTER + 1
327             END STORE;
328
329             PROCEDURE SKIP LINE;
330             BEGIN   INTEGER N;
331                     N:= LAST CHARACTER;
332             SKIP:   IF N ≠ NLCR SYMBOL THEN BEGIN N:= CHARIN;  GOTO SKIP END SKIP
333             END SKIP LINE;
334
335             PROCEDURE READ BLOCK;
336             BEGIN   INTEGER NUMBER OF CHARACTERS ALREADY PACKED, TERMINATOR, BEGINNING OF STRING, N;
337                     REAL PARTLY PACKED WORD;
338
339                     PROCEDURE PACK(CHARACTER); VALUE CHARACTER; INTEGER CHARACTER;
340                     BEGIN   PARTLY PACKED WORD:= PARTLY PACKED WORD * 512 + CHARACTER;
341                             NUMBER OF CHARACTERS ALREADY PACKED:= NUMBER OF CHARACTERS ALREADY PACKED + 1;
342                             IF NUMBER OF CHARACTERS ALREADY PACKED= 3 THEN
343                             BEGIN   STORE(IF PARTLY PACKED WORD > MAXINT THEN PARTLY PACKED WORD - TP27M1 ELSE PARTLY PACKED WORD);
344                                     PARTLY PACKED WORD:= 0;
345                                     NUMBER OF CHARACTERS ALREADY PACKED:= 0
346                             END
347                     END PACK;
348
349                     PARTLY PACKED WORD:= 0;  NUMBER OF CHARACTERS ALREADY PACKED:= 0;  TERMINATOR:= LAST CHARACTER;
350                     BEGINNING OF STRING:= MEMORY POINTER; STORE(0);
351             NEXT:
352                     N:= CHARIN;
353                     IF N ≠ TERMINATOR THEN
354                     BEGIN   PACK(N); GOTO NEXT END;
355                     FOR N:= N WHILE NUMBER OF CHARACTERS ALREADY PACKED ≠ 0 DO PACK(511);
```

```
356                     MEMORY[BEGINNING OF STRING]:= MEMORY POINTER - BEGINNING OF STRING - 1
357             END READ BLOCK;
358
359             COMMENT EDITTING;
360
361             INTEGER INPUT LINE NUMBER, OUTPUT LINE NUMBER, PREVIOUS INPUT LINE NUMBER, START OF BUFFER,
362                     MAXINT, TP18;
363
364             REAL TP27M1;
365
366             BOOLEAN DELETE, EDIT NOTICED, CONTINUATION LINE, EXEC RE;
367
368             PROCEDURE EDIT;
369             BEGIN   INTEGER NUMBER OF CHARACTERS, CHARACTER POINTER, REQUESTED LINE NUMBER;
370                     SWITCH ORDER:=  IDC, IDL, IDN, IDX, IDE,
371                                     ICC, ICL, ICN, ICX, ICE,
372                                     IIC, IIL, ISN, IEC, IIB,
373                                     IRE, ITD, IST;
374
375                     EDIT NOTICED:= FALSE;   GOTO ORDER[TYPE];
376             NEXT INSTRUCTION:
377                     EDIT NOTICED:= TRUE;    GOTO ORDER[TYPE];
378
379             IDC:    DELETE:= TRUE; GOTO IC;
380             ICC:    DELETE:= FALSE;
381             IC:     NUMBER OF CHARACTERS:= MEMORY[MEMORY POINTER];
382                     FOR CHARACTER POINTER:= 1 STEP 1 UNTIL NUMBER OF CHARACTERS DO
383                     BEGIN SEARCH INCLUSIVE(MEMORY[MEMORY POINTER + CHARACTER POINTER]);   EDIT NOTICED:= TRUE   END;
384                     GOTO NEXT INSTRUCTION;
385
386             IDN:    DELETE:= TRUE;   GOTO IN;
387             ICN:    DELETE:= FALSE;
388             IN:     REQUESTED LINE NUMBER:= MEMORY[MEMORY POINTER + 1];
389                     IF REQUESTED LINE NUMBER ≤ INPUT LINE NUMBER THEN
390                     BEGIN   ERROR(↓SPECIFIED LINE ALREADY PROCESSED↓);
391                             GOTO IRE
392                     END BACKWARD MOVE REQUESTED;
393                     FOR INPUT LINE NUMBER:= INPUT LINE NUMBER WHILE REQUESTED LINE NUMBER > INPUT LINE NUMBER DO
394                     SEARCH INCLUSIVE(NLCR SYMBOL);
395                     GOTO NEXT INSTRUCTION;
396
397             IDX:    DELETE:= TRUE;   GOTO IX;
398             ICX:    DELETE:= FALSE;
399             IX:     NUMBER OF CHARACTERS:= MEMORY[MEMORY POINTER];
400                     FOR CHARACTER POINTER:= 1 STEP 1 UNTIL NUMBER OF CHARACTERS DO
401                     BEGIN   PUT(CHARIN);
402                             SEARCH(MEMORY[MEMORY POINTER + CHARACTER POINTER]);
403                             EDIT NOTICED:= TRUE;   CHARACTER IN STOCK:= LAST CHARACTER
404                     END;
405                     GOTO NEXT INSTRUCTION;
406
407             IDE:    DELETE:= TRUE; GOTO IE;
408             ICE:    DELETE:= FALSE;
409             IE:     SEARCH END OF LINE(MEMORY[MEMORY POINTER + 1]);
410                     GOTO NEXT INSTRUCTION;
411
412             IDL:    DELETE:= TRUE; GOTO IL;
413             ICL:    DELETE:= FALSE;
414             IL:     SEARCH LINE;
415                     GOTO NEXT INSTRUCTION;
```

```
416
417                 IIL:    DELETE:= FALSE; PUT(NLCR SYMBOL);
418                 IIC:    DELETE:= FALSE;
419                         NUMBER OF CHARACTERS:= MEMORY[MEMORY POINTER];
420                         FOR CHARACTER POINTER:= 1 STEP 1 UNTIL NUMBER OF CHARACTERS DO
421                         PUT(MEMORY[MEMORY POINTER + CHARACTER POINTER]);
422                         GOTO NEXT INSTRUCTION;
423
424                 ISN:    INPUT LINE NUMBER:= MEMORY[MEMORY POINTER + 1];
425                         GOTO NEXT INSTRUCTION;
426
427                 IEC:    DELETE:= FALSE;
428                         NUMBER OF CHARACTERS:= MEMORY[MEMORY POINTER];
429                         FOR CHARACTER POINTER:= 1 STEP 1 UNTIL NUMBER OF CHARACTERS DO
430                         BEGIN SEARCH(MEMORY[MEMORY POINTER + CHARACTER POINTER]);   EDIT NOTICED:= TRUE END;
431                         GOTO NEXT INSTRUCTION;
432
433                 IIB:    DELETE:= FALSE;
434                         INSERT BLOCK;
435                         GOTO NEXT INSTRUCTION;
436
437                 ITD:    DELETE:= FALSE;
438                         TRANSFER DIRECTLY(MEMORY[MEMORY POINTER + 1]);
439                         GOTO NEXT INSTRUCTION;
440
441                 IST:    DELETE:= FALSE;
442                         NEW LINE;
443                         GOTO EXIT;
444
445                 IRE:    DELETE:= FALSE;   EXEC RE:= TRUE;
446                 COPY REST OF TEXT:
447                         PUT(CHARIN);
448                         GOTO COPY REST OF TEXT
449
450                 END EDIT;
451
452                 INTEGER PROCEDURE TYPE;
453                 BEGIN   MEMORY POINTER:= MEMORY POINTER + MEMORY[MEMORY POINTER] + 1;
454                         TYPE:= MEMORY[MEMORY POINTER];
455                         MEMORY POINTER:= MEMORY POINTER + 1
456                 END TYPE;
457
458                 PROCEDURE PUT(CHARACTER); VALUE CHARACTER;   INTEGER CHARACTER;
459                 IF ~ DELETE THEN
460                 BEGIN   PUNCH SYMBOL(CHARACTER);   PRINT SYMBOL(CHARACTER)  END PUT;
461
462                 PROCEDURE SEARCH INCLUSIVE(CHARACTER); VALUE CHARACTER; INTEGER CHARACTER;
463                 BEGIN   SEARCH(CHARACTER); PUT(CHARACTER) END SEARCH INCLUSIVE;
464
465                 PROCEDURE SEARCH END OF LINE(NUMBER OF NLCRS);   VALUE NUMBER OF NLCRS;   INTEGER NUMBER OF NLCRS;
466                 BEGIN   INTEGER I;
467                         FOR I:= 1 STEP 1 UNTIL NUMBER OF NLCRS DO SEARCH INCLUSIVE(NLCR SYMBOL);
468                         SEARCH(NLCR SYMBOL);   CHARACTER IN STOCK:= NLCR SYMBOL;  INPUT LINE NUMBER:= INPUT LINE NUMBER - 1
469                 END SEARCH END OF LINE;
470
471                 PROCEDURE SEARCH(CHARACTER); VALUE CHARACTER; INTEGER CHARACTER;
472                 BEGIN   INTEGER N;
473                 NEXT:   N:= CHARIN;
474                         IF N ≠ CHARACTER THEN
475                         BEGIN   PUT(N); GOTO NEXT END;
```

```
476                 END SEARCH;
477
478                 PROCEDURE SEARCH LINE;
479                 BEGIN   INTEGER COMPARISON POINTER, END OF STRING, BUFFER POINTER, INPUT, COMPARISON CHARACTER;
480
481                         PROCEDURE DUMP;
482                         BEGIN   INTEGER DUMP POINTER;
483                                 FOR DUMP POINTER:= START OF BUFFER STEP 1 UNTIL BUFFER POINTER - 1 DO
484                                 PUT(MEMORY[DUMP POINTER]);
485                                 COMPARISON POINTER:= MEMORY POINTER + 1;
486                                 BUFFER POINTER:= START OF BUFFER
487                         END DUMP;
488
489                         COMPARISON POINTER:= MEMORY POINTER + 1; END OF STRING:= MEMORY POINTER + MEMORY[MEMORY POINTER];
490                 RETRY:  BUFFER POINTER:= START OF BUFFER;
491                         SEARCH INCLUSIVE(NLCR SYMBOL);
492                 NEXT INPUT:
493                         IF COMPARISON POINTER > END OF STRING THEN GOTO FOUND;
494                         INPUT:= CHARIN;
495                         MEMORY[BUFFER POINTER]:= INPUT; BUFFER POINTER:= BUFFER POINTER + 1;
496                         IF LAYOUT(INPUT) THEN GOTO NEXT INPUT;
497                         COMPARISON CHARACTER:= MEMORY[COMPARISON POINTER]; COMPARISON POINTER:= COMPARISON POINTER + 1;
498                         IF INPUT= NLCR SYMBOL THEN
499                         BEGIN   DUMP;
500                                 GOTO NEXT INPUT
501                         END FAILS BECAUSE OF END OF LINE;
502                         IF INPUT ≠ COMPARISON CHARACTER THEN
503                         BEGIN   DUMP;
504                                 GOTO RETRY
505                         END FAILS BECAUSE OF WRONG CHARACTER;
506                         GOTO NEXT INPUT;
507
508                 FOUND:  DUMP
509                 END SEARCH LINE;
510
511                 PROCEDURE INSERT BLOCK;
512                 BEGIN   INTEGER NUMBER OF PACKED WORDS, WORD POINTER, CHARACTER COUNTER, CHARACTER;
513                         REAL PARTLY PACKED WORD;
514                         NUMBER OF PACKED WORDS:= MEMORY[MEMORY POINTER];
515                         FOR WORD POINTER:= 1 STEP 1 UNTIL NUMBER OF PACKED WORDS DO
516                         BEGIN   PARTLY PACKED WORD:= MEMORY[MEMORY POINTER + WORD POINTER];
517                                 PARTLY PACKED WORD:= (IF 1/PARTLY PACKED WORD < 0
518                                                       THEN PARTLY PACKED WORD + TP27M1
519                                                       ELSE PARTLY PACKED WORD) / TP18;
520                                 FOR CHARACTER COUNTER:= 1, 2, 3 DO
521                                 BEGIN   CHARACTER:= ENTIER(PARTLY PACKED WORD);
522                                         IF CHARACTER ≠ 511 THEN BEGIN  PUT(CHARACTER);   EDIT NOTICED:= TRUE  END;
523                                         PARTLY PACKED WORD:= (PARTLY PACKED WORD - CHARACTER) * 512
524                                 END CHARACTER COUNTER
525                         END WORD POINTER
526                 END INSERT BLOCK;
527
528                 PROCEDURE TRANSFER DIRECTLY(ITEM);   VALUE ITEM;   INTEGER ITEM;
529                 BEGIN   INTEGER SAVE LINE POINTER;
530                         SAVE LINE POINTER:= LINE POINTER;
531                         NEW LINE;   CONTINUATION LINE:= TRUE;
532                         REPSYM(NLCR SYMBOL);
533                         REPTEXT(¢TRANSFERRED DIRECTLY:  ¢);   PRINTS(ITEM);   RESULT(ITEM);
534                         REPSYM(NLCR SYMBOL);
535                         FOR LINE POINTER:= 0 STEP 1 UNTIL SAVE LINE POINTER DO
```

```
536                     LINE[LINE POINTER]:= SPACE SYMBOL;
537                     LINE POINTER:= SAVE LINE POINTER
538             END TRANSFER DIRECTLY;
539
540             COMMENT OUTPUT ORGANIZATION;
541
542             INTEGER LINE POINTER;
543
544             INTEGER ARRAY LINE[0 : 135];
545
546             PROCEDURE PUNCH SYMBOL(CHARACTER);   VALUE CHARACTER;   INTEGER CHARACTER;
547             BEGIN   IF CHARACTER>255 THEN
548                     BEGIN   RESULT(VERTICAL BAR);
549                             CHARACTER:= CHARACTER - 256
550                     END;
551                     IF CHARACTER>127 THEN
552                     BEGIN   RESULT(UNDERLINE);
553                             CHARACTER:= CHARACTER - 128
554                     END;
555                     IF CHARACTER ≠ INTERN END THEN RESULT(IF CHARACTER = INTERN UNDEF THEN UNDEFINED ELSE CHARACTER)
556             END PUNCH SYMBOL;
557
558             PROCEDURE PRINT SYMBOL(CHARACTER);  VALUE CHARACTER;  INTEGER CHARACTER;
559             BEGIN   INTEGER TAB POINTER;
560                     IF CHARACTER= NLCR SYMBOL THEN
561                     BEGIN  NEW LINE;   CONTINUATION LINE:= FALSE   END NLCR SYMBOL ELSE
562                     IF CHARACTER= TAB SYMBOL THEN
563                     BEGIN   TAB POINTER:= (LINE POINTER + 9) ÷ 8 * 8;
564                             IF TAB POINTER = 144 THEN
565                             BEGIN NEW LINE;   CONTINUATION LINE:= TRUE  END OVERFLOW ELSE
566                             BEGIN
567                             SET TAB:
568                                     LINE[LINE POINTER]:= SPACE SYMBOL;
569                                     LINE POINTER:= LINE POINTER + 1;
570                                     IF LINE POINTER ≠ TAB POINTER THEN GOTO SET TAB
571                             END
572                     END TAB SYMBOL ELSE
573                     BEGIN   IF LINE POINTER > 135 THEN BEGIN NEW LINE;   CONTINUATION LINE:= TRUE  END;
574                             LINE[LINE POINTER]:= CHARACTER;
575                             LINE POINTER:= LINE POINTER + 1
576                     END
577             END PRINT SYMBOL;
578
579             PROCEDURE NEW LINE;
580             BEGIN   INTEGER LCNT;
581                     IF CONTINUATION LINE THEN SPACE(5) ELSE
582                     BEGIN   PRINT5(OUTPUT LINE NUMBER);   OUTPUT LINE NUMBER:= OUTPUT LINE NUMBER + 1  END;
583
584                     REPSYM(IF INPUT LINE NUMBER > PREVIOUS INPUT LINE NUMBER + 1 THEN CROSS ELSE SPACE SYMBOL);
585                     PREVIOUS INPUT LINE NUMBER:= INPUT LINE NUMBER;
586
587                     REPSYM(IF EDIT NOTICED THEN STAR ELSE SPACE SYMBOL);
588                     EDIT NOTICED:= FALSE;
589
590                     REPSYM(SPACE SYMBOL);   LINE POINTER:= LINE POINTER - 1;
591                     FOR LCNT:= 0 STEP 1 UNTIL LINE POINTER DO CHAROUT(LINE[LCNT]);
592                     REPSYM(NLCR SYMBOL);
593                     LINE POINTER:= 0
594             END NEW LINE;
595
```

```
596                 PROCEDURE PRINT5(N);   VALUE N;   INTEGER N;
597                 BEGIN   INTEGER SUM, CNT;   REAL X;
598                         SUM:= 0;   CNT:= 5;
599                         X:= N/₁₀4 + 5₁₀-6;
600                 REP:    N:= ENTIER(X);   SUM:= SUM + N;   CNT:= CNT - 1;
601                         IF CNT = 0 THEN REPSYM(N) ELSE
602                         BEGIN   REPSYM(IF SUM = 0 THEN 93 ELSE N);   X:= (X - N) * 10;   GOTO REP   END
603                 END PRINT5;
604
605                 PROCEDURE REPTEXT(STRING);   STRING STRING;
606                 BEGIN   INTEGER I, SYMBOL;
607                          := 0;
608                         FOR SYMBOL:= STRINGSYMBOL(I,STRING) WHILE SYMBOL ≠ 255 DO
609                         BEGIN REPSYM(SYMBOL);   I:= I + 1   END
610                 END REPTEXT;
611
612                 MAXINT:= 2↑26 - 1;   TP27M1:= 2↑27 - 1;   TP18:= 2↑18;
613                 CROSS:= 33;   STAR:= 66; SPACE SYMBOL:= 93; TAB SYMBOL:= 118;
614                 UNDEFINED:= -1;   INTERN UNDEF:= 36;   END:= -4096;   INTERN END:= 63;   STOP:= 511;   CHARACTER IN STOCK:= -1;
615                 NLCR SYMBOL:= LAST CHARACTER:= 119;   NEWPAGE SYMBOL:= 113; UNDERLINE:= 126; VERTICAL BAR:= 127;
616                 DELETE:= FALSE;
617
618                 DC:= CONVERT INSTRUCTION(↑DC↑);
619                 DL:= CONVERT INSTRUCTION(↑DL↑);
620                 DN:= CONVERT INSTRUCTION(↑DN↑);
621                 DX:= CONVERT INSTRUCTION(↑DX↑);
622                 DE:= CONVERT INSTRUCTION(↑DE↑);
623                 CC:= CONVERT INSTRUCTION(↑CC↑);
624                 CL:= CONVERT INSTRUCTION(↑CL↑);
625                 CN:= CONVERT INSTRUCTION(↑CN↑);
626                 CX:= CONVERT INSTRUCTION(↑CX↑);
627                 CE:= CONVERT INSTRUCTION(↑CE↑);
628                 IC:= CONVERT INSTRUCTION(↑IC↑);
629                 IL:= CONVERT INSTRUCTION(↑IL↑);
630                 SN:= CONVERT INSTRUCTION(↑SN↑);
631                 EC:= CONVERT INSTRUCTION(↑EC↑);
632                 IB:= CONVERT INSTRUCTION(↑IB↑);
633                 RE:= CONVERT INSTRUCTION(↑RE↑);
634                 TD:= CONVERT INSTRUCTION(↑TD↑);
635                 ST:= CONVERT INSTRUCTION(↑ST↑);
636                 CO:= CONVERT INSTRUCTION(↑CO↑);
637
638                 ALTER:= MEMORY POINTER:= 0; STORE(0);   PASS 1:= TRUE;   ERRORS DETECTED:= FALSE;
639                 READ OPERATION TAPE;
640                 IF ERRORS DETECTED THEN GOTO EXIT;
641                 START OF BUFFER:= MEMORY POINTER; INPUT LINE NUMBER:= LINE POINTER:= MEMORY POINTER:= PREVIOUS INPUT LINE NUMBER:= 0;
642                 OUTPUT LINE NUMBER:= 1;
643                 PASS 1:= EXEC RE:= FALSE;   CONTINUATION LINE:= TRUE;
644                    REPSYM(NEWPAGE SYMBOL);
645                 EDIT;
646         EXIT:
647
648         END ALTER;
649
650
651         INIT PUNCHLIST;  NLCR COUNT:= 0;   TO PUNCH(114);   TO PUNCH(114);
652         ALTER(RESYMBOL, RESYMBOL, SEP FIRST LINE, REPSYM, AVAILABLE - 1000)
653
654 END
655
```