```
 1 (*$U+,P-,T+,I+*)
 2 (* PASCAL-M COMPILER *)
 3 (* COPYRIGHT 1977, MARK D. RUSTAD *)
 4
 5 PROGRAM PASCALCOMPILER(INPUT,OUTPUT);
 6
 7 CONST DISPLIMIT=15; MAXLEVEL=12; MAXADDR=32767;
 8   INTSIZE=2; REALSIZE=4;
 9   CHARSIZE=1; BOOLSIZE=1; SETSIZE=8; PTRSIZE=2;
10   STRGLGTH=8; MAXINT=32767; SETMAX=58;
11   LCAFTERMARKSTACK=6;
12   MAXCHCNT = 120;     (* MAXIMUM INPUT LINE LENGTH *)
13   (* ASCII CHARACTER CONSTANTS *)
14   AA=65; AF=70; AZ=90; A0=48; A9=57; APLUS=43; AMINUS=45; ASTAR=42;
15   ASLASH=47; ALPAREN=40; ARPAREN=41; ADOLLAR=36; AEQUAL=61;
16   ASPACE=32; ACOMMA=44; APERIOD=46; ADQUOTE=34; ALBRACK=91;
17   ARBRACK=93; ACOLON=58  AARROW=94; ALESS=60; AGREATER=62; ASEMI=59;
18
19 TYPE  (* DESCRIBING: *)
20
21   (* BASIC SYMBOLS*)
22   (*****************)
23   SYMBOL=(IDENT,INTCONST,REALCONST,STRINGCONST,NOTSY,MULOP,
24     ADDOP,RELOP,LPARENT,RPARENT,LBRACK,RBRACK,COMMA,
25     SEMICOLON,PERIOD,ARROW,COLON,BECOMES,CONSTSY,TYPESY,
26     VARSY,FUNCSY,PROGSY,PROCSY,SETSY,PACKEDSY,ARRAYSY,
27     RECORDSY,EXTERNSY,FORWARDSY,BEGINSY,IFSY,CASESY,REPEATSY,
28     WHILESY,FORSY,ENDSY,ELSESY,UNTILSY,OFSY,DOSY,TOSY,
29     DOWNTOSY,THENSY,OTHERSY);
30
31   OPERATOR=(MUL,RDIV,ANDOP,IDIV,IMOD,PLUS,MINUS,OROP,LTOP,
32     LEOP,GEOP,GTOP,NEOP,EQOP,INOP,NOOP);
33
34   SETOFSYS=SET OF SYMBOL;
35   INTSET=SET OF 0..SETMAX;
36
37   (* CONSTANTS *)
38   (*************)
39   CSP=^CONSTANT;
40   CONSTANT = RECORD CASE INTEGER OF
41                (*SET*) 0: (PVAL: INTSET);
42                (*STRING*) 1: (SLGTH: 0..STRGLGTH; SPTR: CSP;
43                  SVAL: PACKED ARRAY [1..STRGLGTH] OF 0..127)
44              END;
45   VALU = RECORD CASE INTEGER OF
46            0: (IVAL: INTEGER);
47            1: (VALP: CSP)
48          END;
49
50   (* DATA STRUCTURES *)
51   (*******************)
52   LEVRANGE=0..MAXLEVEL; ADDRRANGE=0..MAXADDR;
53   STRUCTFORM=(SCALAR,SUBRANGE,POINTER,POWER,ARRAYS,RECORDS);
54   DECLKIND=(STANDARD,DECLARED);
55   STP=^STRUCTURE; CTP=^IDENTIFIER;
56   STRUCTURE=PACKED RECORD
57                 SIZE: ADDRRANGE;
58                 STYPE: STP;
59                 CASE FORM: STRUCTFORM OF
60                   SUBRANGE: (MIN,MAX: VALU);
61                   ARRAYS: (INXTYPE: STP);
62                   RECORDS: (FSTFLD: CTP)
```

```
                END;
64
65    (* NAMES *)
66    (********)
67    IDCLASS=(TYPES,KONST,VARS,FIELD,PROC,FUNC);
68    SETOFIDS=SET OF IDCLASS;
69    IDKIND=(ACTUAL,FORMAL);
70    ALPHA=PACKED ARRAY [1..8] OF CHAR;
71    IDENTIFIER=PACKED RECORD
72                  NAME: ALPHA; LLINK,RLINK: CTP;
73                  IDTYPE: STP; NEXT: CTP;
74                  CASE KLASS: IDCLASS OF
75                    KONST: (VALUES: VALU);
76                    VARS: (VKIND: IDKIND; VLEV: LEVRANGE;
77                      VADDR: ADDRRANGE);
78                    PROC,FUNC:
79                      (CASE PFDECKIND: DECLKIND OF
80                         STANDARD: (KEY: 1..8);
81                         DECLARED: (PFLEV: LEVRANGE; PFNAME: INTEGER;
82                           FORWDECL,EXTERNL: BOOLEAN))
83                END;
84    DISPRANGE=0..DISPLIMIT;
85
86    (* EXPRESSIONS *)
87    (***************)
88    ATTRKIND=(CST,VARBL,EXPR);
89    VACCESS=(DRCT,INDRCT);
90    ATTR=RECORD TYPTR: STP;
91             CASE KIND: ATTRKIND OF
92               CST: (CVAL: VALU);
93               VARBL: (ACCESS: VACCESS; LEVEL: LEVRANGE;
94                 DPLMT: ADDRRANGE)
95           END;
96
97
98  (*-------------------------------------------------------*)
99
100
101 VAR
102   (* RETURNED BY SOURCE PROGRAM SCANNER INSYMBOL: *)
103   SY: SYMBOL;        (* LAST SYMBOL *)
104   OP: OPERATOR;      (* CLASSIFICATION OF LAST SYMBOL *)
105   VAL: VALU;         (* VALUE OF LAST CONSTANT *)
106   LGTH: INTEGER;     (* LENGTH OF LAST STRING CONSTANT *)
107   ID: ALPHA;         (* LAST IDENTIFIER (POSSIBLY TRUNCATED) *)
108   KK: 0..8;          (* NR OF CHARS IN LAST IDENTIFIER *)
109   CH: CHAR;          (* LAST CHARACTER *)
110   ICH: 0..127;       (* INTEGER OF LAST CHARACTER IN ASCII *)
111   LINELENGTH,CHCNT: 0..MAXCHCNT;
112   LINE: PACKED ARRAY [1..MAXCHCNT] OF CHAR; (* LAST SOURCE LINE *)
113   LINECOUNT:INTEGER;
114
115   (* COUNTERS *)
116   (************)
117   LC,IC: ADDRRANGE;(* DATA LOCATION AND INSTRUCTION COUNTERS *)
118   ICN,SUMCHECK: INTEGER;
119   CODEBUF: ARRAY [0..29] OF 0..255;
120   NPROC: INTEGER;  (* NEXT PROCEDURE NUMBER *)
121   I,MXINT10: INTEGER;
122
123   (* SWITCHES *)
124   (************)
```

```
125   DP,                (* DECLARATION PART *)
126   PRTERR: BOOLEAN; (* TO ALLOW FORWARD REF. IN POINTER TYPE
127                             DECLARATION BY SUPPRESSING ERR MSG *)
128   PFCTTEST: BOOLEAN; (* DETECT FUNCTION CALLS IN PARAMETERLISTS *)
129    ERRFLAG : BOOLEAN ; (* COMPILER ERROR FLAG *)
130
131   (* POINTERS: *)
132   (*************)
133   INTPTR,CHARPTR,
134   BOOLPTR,NILPTR: STP; (* POINTERS TO STD IDS *)
135   UTYPPTR,UCSTPTR,UVARPTR,
136   UFLDPTR,UPRCPTR,UFCTPTR, (* POINTERS FOR UNDECL. IDS *)
137   FWPTR: CTP;       (* HEAD OF CHAIN OF FORW DECL TYPE IDS *)
138   PROGPTR: CTP;  (* POINTER TO PROGRAM *)
139
140   (* BOOKKEEPING OF DECLARATION LEVELS: *)
141   (**************************************)
142   LEVEL: LEVRANGE; (* CURRENT STATIC LEVEL *)
143   DISX,              (* LEVEL OF LAST ID SEARCHED BY SEARCHID *)
144   TOP: DISPRANGE;  (* TOP OF DISPLAY *)
145   SAVETOP: DISPRANGE;  (* SAVE TOP OF DISPLAY *)
146   DISPLAY: ARRAY [DISPRANGE] OF CTP;
147
148   (* ERROR MESSAGES: *)
149   (*******************)
150   ERRINX: 0..10;    (* NR OF ERRORS IN CURRENT SOURCE LINE *)
151   ERRLIST:
152     ARRAY [1..10] OF
153       PACKED RECORD POS: 1..132;
154                     NMR: 1..401
155              END;
156
157   (* EXPRESSION COMPILATION: *)
158   (***************************)
159   GATTR: ATTR;      (* DESCRIBES THE EXPRESSION BEING COMPILED *)
160
161   (* STRUCTURED CONSTANTS: *)
162   (*************************)
163   CONSTBEGSYS,SIMPTYPEBEGSYS,TYPEBEGSYS,BLOCKBEGSYS,SELECTSYS,
164   FACBEGSYS,STATBEGSYS,TYPEDELS: SETOFSYS;
165   RW: ARRAY [0..31(*NO RES WDS*)] OF ALPHA;
166   FRW: ARRAY [0..8] OF 0..32 (*NO RES WDS*);
167   RSY: ARRAY [0..31(*NO RES WDS*)] OF SYMBOL;
168   SSY: ARRAY [0..127] OF SYMBOL;
169   ROP: ARRAY [0..31(*NO RES WDS*)] OF OPERATOR;
170   SOP: ARRAY [0..127] OF OPERATOR;
171   NA: ARRAY [0..12] OF ALPHA;
172
173
174 (*-----------------------------------------------------------*)
175 PROCEDURE TTYIN;EXTERN = $00C0;
176 PROCEDURE SETHSR;EXTERN = $00C8;
177 PROCEDURE TTYOUT;EXTERN = $00D0;
178 PROCEDURE SETHSP;EXTERN= $00D8;
179 PROCEDURE BEGINLINE;
180 BEGIN
181   IF EOF THEN WRITELN(' PREMATURE END OF SOURCE FILE');
182   LINELENGTH:=1; READ(LINE[LINELENGTH]);
183   LINECOUNT:=LINECOUNT+1;
184   WHILE NOT EOLN AND (LINELENGTH<MAXCHCNT) DO
185     BEGIN LINELENGTH:=LINELENGTH+1;
186       READ(LINE[LINELENGTH])
```

```
      END;
  IF NOT EOLN THEN READLN; CHCNT :=0
END (* BEGINLINE *);

PROCEDURE ENDLINE;
VAR LASTPOS,FREEPOS,CURRPOS,CURRNMR,F,K: INTEGER;
BEGIN
   IF ERRINX>0 THEN   (* OUTPUT ERROR MESSAGES *)
     BEGIN
  TTYOUT;
   WRITE(≡ ≡,LINECOUNT:4);
       FOR K:=1 TO LINELENGTH DO WRITE(LINE[K]);
       WRITELN;
       WRITE(≡ ****≡);
       LASTPOS:=0; FREEPOS:=1;
       FOR K:=1 TO ERRINX DO
         BEGIN
           CURRPOS:=ERRLIST[K].POS; CURRNMR:=ERRLIST[K].NMR;
           IF CURRPOS=LASTPOS THEN WRITE(≡,≡)
           ELSE
             BEGIN
               WHILE FREEPOS<CURRPOS DO
                 BEGIN WRITE(≡ ≡); FREEPOS:=FREEPOS+1 END;
               WRITE(≡↑≡);
               LASTPOS:=CURRPOS
             END;
           IF CURRNMR<10 THEN F:=1
           ELSE IF CURRNMR<100 THEN F:=2
             ELSE F:=3;
           WRITE(CURRNMR:F);
           FREEPOS:=FREEPOS+F+1
         END;
       WRITELN; ERRINX:=0
     END;
SETHSP
END (* ENDLINE *);

PROCEDURE ERROR(FERRNR: INTEGER);
BEGIN
    ERRFLAG := TRUE ;
  IF ERRINX>=9 THEN
    BEGIN ERRLIST[10].NMR:=255; ERRINX:=10 END
  ELSE
    BEGIN ERRINX:=ERRINX+1;
      ERRLIST[ERRINX].NMR:=FERRNR
    END;
  ERRLIST[ERRINX].POS:=CHCNT
END (* ERROR *);

FUNCTION STRINGSIZE(LVP: CSP): INTEGER;
VAR N: INTEGER;
BEGIN N:=0;
  REPEAT
    N:=N+LVP↑.SLGTH; LVP:=LVP↑.SPTR
  UNTIL LVP=NIL;
  STRINGSIZE:=N
END (* STRINGSIZE *);

PROCEDURE INSYMBOL;
(* READ NEXT BASIC SYMBOL OF SOURCE PROGRAM AND RETURN ITS
   DESCRIPTION IN THE GLOBAL VARIABLES SY, OP, ID, VAL AND
   LGTH *)
```

```
    VAR I,K: INTEGER;
      STRING: PACKED ARRAY [1..STRGLGTH] OF 0..127;
      LVP,LVP1: CSP; TEST,COMNT: BOOLEAN;
      QUOTECHAR: CHAR;

      PROCEDURE NEXTCH;
      BEGIN
        IF CHCNT>LINELENGTH THEN
          BEGIN ENDLINE; BEGINLINE END;
        CHCNT:=CHCNT+1;
        IF CHCNT>LINELENGTH THEN CH:=' '
        ELSE CH:=LINE[CHCNT];
        ICH:=ORD(CH)
      END (* NEXTCH *);

    BEGIN (* INSYMBOL *)
      REPEAT COMNT:=TRUE;
        WHILE CH=' ' DO NEXTCH;
        IF (ICH>=AA) AND (ICH<=AZ) THEN
          BEGIN KK:=0; ID:='          ';
            REPEAT
              IF KK<8 THEN
                BEGIN KK:=KK+1; ID[KK]:=CH END;
              NEXTCH
            UNTIL NOT(((ICH>=A0)AND(ICH<=A9))OR((ICH>=AA)AND(ICH<=AZ)));
            SY:=IDENT; OP:=NOOP; I:=FRW[KK-1];
            WHILE I<=FRW[KK]-1 DO
              BEGIN
                IF RW[I]=ID THEN
                  BEGIN SY:=RSY[I]; OP:=ROP[I] END;
                I:=I+1
              END
          END
        ELSE
          IF (ICH>=A0) AND (ICH<=A9) THEN
            BEGIN OP:=NOOP; TEST:=FALSE;
              K:=0;
              REPEAT
                IF K<=MXINT10 THEN
                  K:=K*10+ICH-A0
                ELSE TEST:=TRUE;
                NEXTCH
              UNTIL (ICH<A0) OR (ICH>A9);
              IF TEST THEN ERROR(203);
              VAL.IVAL:=K; SY:=INTCONST
            END
          ELSE
            IF (CH='''') OR (CH='"') THEN
              BEGIN LGTH:=0; SY:=STRINGCONST; OP:=NOOP; LVP:=NIL;
                QUOTECHAR:=CH;
                REPEAT
                  REPEAT
                    IF CHCNT>LINELENGTH THEN ERROR(202);
                    NEXTCH; LGTH:=LGTH+1;
                    IF LGTH>STRGLGTH THEN
                      BEGIN NEW(LVP1);
                        IF LVP<>NIL THEN LVP^.SPTR:=LVP1
                        ELSE VAL.VALP:=LVP1;
                        LVP:=LVP1;
                        LVP^.SVAL:=STRING; LVP^.SLGTH:=STRGLGTH;
                        LVP^.SPTR:=NIL;
                        LGTH:=1
```

```
311                 END;
312               STRING[LGTH]:=ICH
313             UNTIL CH=QUOTECHAR;
314             NEXTCH
315           UNTIL CH<>QUOTECHAR;
316           LGTH:=LGTH-1;   (* NOW LGTH=NR CHARS IN STRING *)
317           IF (LGTH=1) AND (LVP=NIL) THEN VAL.IVAL:=STRING[1]
318           ELSE
319             IF LGTH<>0 THEN
320               BEGIN NEW(LVP1);
321                 IF LVP<>NIL THEN LVP↑.SPTR:=LVP1 ELSE VAL.VALP:=LVP1;
322                 LVP1↑.SVAL:=STRING; LVP1↑.SLGTH:=LGTH;
323                 LVP1↑.SPTR:=NIL;
324                 LGTH:=STRINGSIZE(VAL.VALP)
325               END;
326         END
327       ELSE
328         IF CH=':' THEN
329           BEGIN OP:=NOOP; NEXTCH;
330             IF CH='=' THEN
331               BEGIN SY:=BECOMES; NEXTCH END
332             ELSE SY:=COLON
333           END
334         ELSE
335           IF CH='.' THEN
336             BEGIN OP:=NOOP; NEXTCH;
337               IF CH='.' THEN
338                 BEGIN SY:=COLON; NEXTCH END
339               ELSE SY:=PERIOD
340             END
341           ELSE
342             IF CH='<' THEN
343               BEGIN NEXTCH; SY:=RELOP;
344                 IF CH='=' THEN
345                   BEGIN OP:=LEOP; NEXTCH END
346                 ELSE
347                   IF CH='>' THEN
348                     BEGIN OP:=NEOP; NEXTCH END
349                   ELSE OP:=LTOP
350               END
351             ELSE
352               IF CH='>' THEN
353                 BEGIN NEXTCH; SY:=RELOP;
354                   IF CH='=' THEN
355                     BEGIN OP:=GEOP; NEXTCH END
356                   ELSE OP:=GTOP
357                 END
358               ELSE
359                 IF CH='(' THEN
360                   BEGIN NEXTCH;
361                     IF CH='*' THEN
362                       BEGIN NEXTCH;
363                         REPEAT
364                           WHILE CH<>'*' DO NEXTCH;
365                           NEXTCH
366                         UNTIL CH=')';
367                         NEXTCH; COMNT:=FALSE
368                       END
369                     ELSE BEGIN SY:=LPARENT; OP:=NOOP END
370                   END
371                 ELSE
372                   IF CH='$' THEN
```

```
373                         BEGIN OP:=NOOP; K:=0;
374                           REPEAT NEXTCH;
375                             IF (ICH>=A0) AND (ICH<=A9) THEN
376                               I:=ICH-A0
377                             ELSE
378                               IF (ICH>=AA) AND (ICH<=AF) THEN
379                                 I:=ICH-AA+10
380                               ELSE I:=-1;
381                             IF I>=0 THEN K:=16*K+I;
382                           UNTIL I<0;
383                           VAL.IVAL:=K; SY:=INTCONST
384                         END
385                       ELSE
386                         BEGIN SY:=SSY[ICH]; OP:=SOP[ICH]; NEXTCH;
387                           IF SY=OTHERSY THEN ERROR(399)
388                         END;
389     UNTIL COMNT
390   END (* INSYMBOL *);
391
392   PROCEDURE ENTERID(FCP: CTP);
393   (* ENTER ID POINTED AT BY FCP INTO THE NAME-TABLE,
394      WHICH ON EACH DECLARATION LEVEL IS ORGANIZED AS
395      AN UNBALANCED BINARY TREE *)
396   VAR NAM: ALPHA; LCP,LCP1: CTP; LLEFT: BOOLEAN;
397   BEGIN NAM:=FCP↑.NAME;
398     LCP:=DISPLAY[TOP];
399     IF LCP=NIL THEN
400       DISPLAY[TOP]:=FCP
401     ELSE
402       BEGIN
403         REPEAT LCP1:=LCP;
404           IF LCP↑.NAME=NAM THEN  (* NAME CONFLICT, FOLLOW RIGHT LINK *)
405             BEGIN ERROR(101); LCP:=LCP↑.RLINK; LLEFT:=FALSE END
406           ELSE
407             IF LCP↑.NAME<NAM THEN
408               BEGIN LCP:=LCP↑.RLINK; LLEFT:=FALSE END
409             ELSE BEGIN LCP:=LCP↑.LLINK; LLEFT:=TRUE END
410         UNTIL LCP=NIL;
411         IF LLEFT THEN LCP1↑.LLINK:=FCP ELSE LCP1↑.RLINK:=FCP
412       END;
413     FCP↑.LLINK:=NIL; FCP↑.RLINK:=NIL
414   END (* ENTERID *);
415
416   PROCEDURE SEARCHSECTION(FCP: CTP; VAR FCP1: CTP);
417   (* TO FIND RECORD FIELDS AND FORWARD DECLARED PROCEDURE ID=S
418      --> PROCEDURE PROCEDUREDECLARATION
419      --> PROCEDURE SELECTOR *)
420   BEGIN
421     FCP1:=NIL;
422     WHILE (FCP<>NIL) AND (FCP1=NIL) DO
423       IF FCP↑.NAME<>ID THEN
424         IF FCP↑.NAME<ID THEN FCP:=FCP↑.RLINK
425         ELSE FCP:=FCP↑.LLINK
426       ELSE FCP1:=FCP
427   END (* SEARCHSECTION *);
428
429   PROCEDURE SEARCHID(FIDCLS: SETOFIDS; VAR FCP: CTP);
430   VAR LCP: CTP;
431   BEGIN
432     DISX:=TOP+1; FCP:=NIL;
433     WHILE (DISX>0) AND (FCP=NIL) DO
434       BEGIN DISX:=DISX-1;
```

```
43        LCP:=DISPLAY[DISX];
436       WHILE (LCP<>NIL) AND (FCP=NIL) DO
437         IF LCP↑.NAME=ID THEN
438           IF LCP↑.KLASS IN FIDCLS THEN FCP:=LCP
439           ELSE ;
440             BEGIN IF PRTERR THEN ERROR(103);
441               LCP:=LCP↑.RLINK
442             END
443         ELSE
444           IF LCP↑.NAME<ID THEN
445             LCP:=LCP↑.RLINK
446           ELSE LCP:=LCP↑.LLINK
447     END;
448 (* SEARCH NOT SUCCESSFUL; SUPPRESS ERROR MESSAGE IN CASE
449    OF FORWARD REFERENCED TYPE ID IN POINTER TYPE DEFINITION
450    --> PROCEDURE SIMPLETYPE *)
451   IF PRTERR AND (FCP=NIL) THEN
452     BEGIN ERROR(104);
453       (* TO AVOID RETURNING NIL, REFERENCE AN ENTRY FOR
454          UNDECLARED ID OF APPROPRIATE CLASS
455          --> PROCEDURE ENTERUNDECL *)
456       IF TYPES IN FIDCLS THEN FCP:=UTYPPTR
457       ELSE
458         IF VARS IN FIDCLS THEN FCP:=UVARPTR
459         ELSE
460           IF FIELD IN FIDCLS THEN FCP:=UFLDPTR
461           ELSE
462             IF KONST IN FIDCLS THEN FCP:=UCSTPTR
463             ELSE
464               IF PROC IN FIDCLS THEN FCP:=UPRCPTR
465               ELSE FCP:=UFCTPTR;
466     END;
467 END (* SEARCHID *);
468
469 PROCEDURE GETBOUNDS(FSP: STP; VAR FMIN,FMAX: INTEGER);
470 (* GET INTERNAL BOUNDS OF SUBRANGE OR SCALAR TYPE *)
471 (* ASSUME (FSP<>NIL) AND (FSP↑.FORM<=SUBRANGE) AND (FSP<>INTPTR)
472    AND NOT COMPTYPES(REALPTR,FSP) *)
473 BEGIN
474   IF FSP↑.FORM=SUBRANGE THEN
475     BEGIN FMIN:=FSP↑.MIN.IVAL; FMAX:=FSP↑.MAX.IVAL END
476   ELSE
477     BEGIN FMIN:=0;
478       IF FSP=CHARPTR THEN FMAX:=127
479         ELSE FMAX:=0
480     END
481 END (* GETBOUNDS *);
482
483 PROCEDURE HEXOUT(WORD: INTEGER);
484 VAR K: INTEGER;
485 BEGIN SUMCHECK:=SUMCHECK+WORD; K:=WORD DIV 16; WORD:=WORD MOD 16;
486   IF K<10 THEN WRITE(CHR(ORD('0') + K))
487   ELSE WRITE(CHR(ORD('A')+K-10));
488   IF WORD<10 THEN WRITE(CHR(ORD('0')+WORD))
489   ELSE WRITE(CHR(ORD('A')+WORD-10))
490 END (* HEXOUT *);
491
492 PROCEDURE WRITEOUT;
493 VAR I: INTEGER;
494 BEGIN (* WRITEOUT *)
495   IF ICN<>0 THEN
496     BEGIN WRITE(' P1'); SUMCHECK:=0; HEXOUT(ICN+1);
```

```
497     FOR I:=0 TO ICN-1 DO HEXOUT(CODEBUF[I]);
498       HEXOUT((16383-SUMCHECK) MOD 256); WRITELN
499     END;
500   IC:=IC+ICN; ICN:=0
501 END (* WRITEOUT *);
502
503 PROCEDURE BLOCK(FSYS: SETOFSYS; FSY: SYMBOL; FPROCP: CTP);
504 TYPE OPRANGE=0..255;
505 VAR LSY: SYMBOL; TEST: BOOLEAN;
506
507   PROCEDURE BYTEGEN(BYTE: INTEGER);
508   BEGIN
509     IF ICN=30 THEN WRITEOUT;
510     CODEBUF[ICN]:=BYTE;
511     ICN:=ICN+1
512   END (* BYTEGEN *);
513
514   PROCEDURE ADDRGEN(ADDR: INTEGER);
515   BEGIN
516     BYTEGEN(ADDR DIV 256); BYTEGEN(ADDR MOD 256)
517   END (* ADDRGEN *);
518
519   PROCEDURE GENUJPENT(FOP: OPRANGE; FP2: INTEGER);
520   BEGIN
521     BYTEGEN(FOP); ADDRGEN(FP2)
522   END (* GENUJPENT *);
523
524   PROCEDURE PLANTADDR(LOC,VAL: INTEGER);
525   BEGIN
526     IF VAL<>0 THEN
527       IF (LOC>=IC)AND(LOC-IC<ICN) THEN
528         BEGIN LOC:=LOC-IC; CODEBUF[LOC]:=VAL DIV 256;
529           CODEBUF[LOC+1]:=VAL MOD 256
530         END
531       ELSE
532         BEGIN WRITEOUT; WRITE(# P2#); SUMCHECK:=0;
533           HEXOUT(LOC DIV 256); HEXOUT(LOC MOD 256);
534           HEXOUT(VAL DIV 256); HEXOUT(VAL MOD 256);
535           HEXOUT((16383-SUMCHECK) MOD 256); WRITELN
536         END
537   END (* PLANTADDR *);
538
539   PROCEDURE SKIP(FSYS: SETOFSYS);
540   (* SKIP INPUT STRING UNTIL RELEVANT SYMBOL FOUND *)
541   BEGIN WHILE NOT(SY IN FSYS) DO INSYMBOL
542   END (* SKIP *);
543
544   PROCEDURE TEST1(FSYS: SETOFSYS; N: INTEGER);
545   BEGIN
546     IF NOT(SY IN FSYS) THEN
547       BEGIN ERROR(N); SKIP(FSYS) END
548   END (* TEST1 *);
549
550   PROCEDURE TEST2(FSYS: SETOFSYS; N: INTEGER; S2: SETOFSYS);
551   BEGIN
552     IF NOT(SY IN FSYS) THEN
553       BEGIN ERROR(N); SKIP(FSYS+S2) END
554   END (* TEST2 *);
555
556   PROCEDURE INTEST(SYM: SYMBOL; N: INTEGER);
557   BEGIN
558     IF SY=SYM THEN INSYMBOL ELSE ERROR(N)
```

```
    END (* INTEST *);

    PROCEDURE CONSTANT(FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU);
    TYPE SIGNS = (NONE,POS,NEG);
    VAR LSP: STP; LCP: CTP; SIGN: SIGNS;
        LVP: CSP; I: 2..STRGLGTH;
    BEGIN LSP:=NIL; FVALU.IVAL:=0;
      TEST2(CONSTBEGSYS,50,FSYS);
      IF SY IN CONSTBEGSYS THEN
        BEGIN
          IF SY=STRINGCONST THEN
            BEGIN
              IF LGTH=1 THEN LSP:=CHARPTR
              ELSE
                BEGIN
                  NEW(LSP);
                   LSP↑.STYPE:=CHARPTR; LSP↑.INXTYPE:=NIL;
                  LSP↑.SIZE:=STRINGSIZE(VAL.VALP); LSP↑.FORM:=ARRAYS;
                END;
              FVALU:=VAL; INSYMBOL
            END
          ELSE
            BEGIN
              SIGN:=NONE;
              IF (SY=ADDOP) AND (OP IN [PLUS,MINUS]) THEN
                BEGIN IF OP=PLUS THEN SIGN:=POS ELSE SIGN:=NEG;
                  INSYMBOL
                END;
              IF SY=IDENT THEN
                BEGIN SEARCHID([KONST],LCP);
                  LSP:=LCP↑.IDTYPE; FVALU:=LCP↑.VALUES;
                  IF SIGN<>NONE THEN
                    IF LSP=INTPTR THEN
                      BEGIN IF SIGN=NEG THEN FVALU.IVAL:=-FVALU.IVAL END
                      ELSE ERROR(105);
                  INSYMBOL;
                END
              ELSE
                IF SY=INTCONST THEN
                  BEGIN IF SIGN=NEG THEN VAL.IVAL:=-VAL.IVAL;
                    LSP:=INTPTR; FVALU:=VAL; INSYMBOL
                  END
                ELSE
                    BEGIN ERROR(106); SKIP(FSYS) END
            END;
          TEST1(FSYS,6)
        END;
      FSP:=LSP
    END (* CONSTANT *);

    FUNCTION COMPTYPES(FSP1,FSP2: STP): BOOLEAN;
    (* DECIDE WHETHER STRUCTURES POINTED AT BY FSP1 AND FSP2 ARE
       COMPATIBLE *)
    BEGIN
      IF FSP1=FSP2 THEN COMPTYPES:=TRUE
      ELSE
        IF (FSP1<>NIL) AND (FSP2<>NIL) THEN
          IF FSP1↑.FORM=FSP2↑.FORM THEN
            CASE FSP1↑.FORM OF
              SCALAR,RECORDS: COMPTYPES:=FALSE;
              SUBRANGE,POINTER,POWER:
                COMPTYPES:=COMPTYPES(FSP1↑.STYPE,FSP2↑.STYPE);
```

```
            ARRAYS:
              COMPTYPES:=COMPTYPES(FSP1↑.STYPE,FSP2↑.STYPE) AND
                (FSP1↑.SIZE=FSP2↑.SIZE)
          END
        ELSE (* FSP1↑.FORM<>FSP2↑.FORM *)
          IF FSP1↑.FORM=SUBRANGE THEN
            COMPTYPES:=COMPTYPES(FSP1↑.STYPE,FSP2)
          ELSE
            IF FSP2↑.FORM=SUBRANGE THEN
              COMPTYPES:=COMPTYPES(FSP1,FSP2↑.STYPE)
            ELSE COMPTYPES:=FALSE
      ELSE COMPTYPES:=TRUE
  END (* COMPTYPES *);

  FUNCTION STRING(FSP: STP): BOOLEAN;
  BEGIN STRING:=FALSE;
    IF FSP<>NIL THEN
      IF FSP↑.FORM=ARRAYS THEN
        IF COMPTYPES(FSP↑.STYPE,CHARPTR) THEN STRING:=TRUE
  END (* STRING *);

  PROCEDURE TYP(FSYS: SETOFSYS; VAR FSP: STP; VAR FSIZE: ADDRRANGE);
  VAR LSP,LSP1,LSP2: STP; OLDTOP: DISPRANGE; LCP: CTP;
      LSIZE,DISPL: ADDRRANGE; LMIN,LMAX: INTEGER;

    PROCEDURE SIMPLETYPE(FSYS:SETOFSYS;VAR FSP:STP;VAR FSIZE:ADDRRANGE);
    VAR LSP,LSP1: STP; LCP,LCP1: CTP; TTOP: DISPRANGE;
        LCNT: INTEGER; LVALU: VALU;
    BEGIN FSIZE:=1;
      TEST2(SIMPTYPEBEGSYS,1,FSYS);
      IF SY IN SIMPTYPEBEGSYS THEN
        BEGIN
          IF SY=LPARENT THEN
            BEGIN TTOP:=TOP; TOP:=SAVETOP;
              NEW(LSP); LSP↑.FORM:=SCALAR;
              LCP1:=NIL; LCNT:=0;
              REPEAT INSYMBOL;
                IF SY=IDENT THEN
                  BEGIN NEW(LCP); LCP↑.NAME:=ID; LCP↑.IDTYPE:=LSP;
                    LCP↑.NEXT:=LCP1; LCP↑.VALUES.IVAL:=LCNT;
                    LCP↑.KLASS:=KONST;
                    ENTERID(LCP); LCNT:=LCNT+1; LCP1:=LCP; INSYMBOL
                  END
                ELSE ERROR(2);
                TEST1(FSYS+[COMMA,RPARENT],6);
              UNTIL SY<>COMMA;
              LSP↑.FSTFLD:=LCP1; TOP:=TTOP;
              IF LCNT<257 THEN LSP↑.SIZE:=1 ELSE LSP↑.SIZE:=INTSIZE;
              INTEST(RPARENT,4)
            END
          ELSE
          IF SY=IDENT THEN
            BEGIN SEARCHID([TYPES,KONST],LCP);
              INSYMBOL;
              IF LCP↑.KLASS=KONST THEN
                BEGIN NEW(LSP);
                  LSP↑.STYPE:=LCP↑.IDTYPE; LSP↑.FORM:=SUBRANGE;
                  IF STRING(LSP↑.STYPE) THEN
                    BEGIN ERROR(148); LSP↑.STYPE:=NIL END;
                  LSP↑.MIN:=LCP↑.VALUES;
                  LSP↑.SIZE:=LCP↑.IDTYPE↑.SIZE;
                  INTEST(COLON,5);
```

```
                    CONSTANT(FSYS,LSP1,LVALU);
                    LSP↑.MAX:=LVALU;
                    IF LSP↑.STYPE<>LSP1 THEN ERROR(107)
                  END
                ELSE
                  LSP:=LCP↑.IDTYPE
              END (* SY=IDENT *)
            ELSE
              BEGIN NEW(LSP); LSP↑.FORM:=SUBRANGE;
                CONSTANT(FSYS+[COLON],LSP1,LVALU);
                IF STRING(LSP1) THEN
                  BEGIN ERROR(148); LSP1:=NIL END;
                LSP↑.STYPE:=LSP1; LSP↑.MIN:=LVALU; LSP↑.SIZE:=INTSIZE;
                INTEST(COLON,5);
                CONSTANT(FSYS,LSP1,LVALU);
                LSP↑.MAX:=LVALU;
                IF LSP↑.STYPE<>LSP1 THEN ERROR(107)
              END;
            IF LSP<>NIL THEN
              IF LSP↑.FORM=SUBRANGE THEN
                IF LSP↑.MIN.IVAL>LSP↑.MAX.IVAL THEN ERROR(102);
            FSP:=LSP;
            TEST1(FSYS,6)
          END
        ELSE FSP:=NIL;
        IF LSP<>NIL THEN FSIZE:=LSP↑.SIZE
      END (* SIMPLETYPE *);

      PROCEDURE FIELDLIST(FSYS: SETOFSYS);
      VAR LCP,LCP1,NXT,NXT1: CTP; LSP,LSP3: STP;
        MINSIZE,MAXSIZE,LSIZE: ADDRRANGE; LVALU: VALU;
      BEGIN NXT1:=NIL; LSP:=NIL;
        TEST2([IDENT,CASESY],19,FSYS);
        WHILE SY=IDENT DO
          BEGIN NXT:=NXT1;
            REPEAT
              IF SY=IDENT THEN
                BEGIN NEW(LCP);
                  LCP↑.NAME:=ID; LCP↑.IDTYPE:=NIL; LCP↑.NEXT:=NXT;
                  LCP↑.KLASS:=FIELD;
                  NXT:=LCP;
                  ENTERID(LCP);
                  INSYMBOL
                END
              ELSE ERROR(2);
              TEST2([COMMA,COLON],6,FSYS+[SEMICOLON,CASESY]);
              TEST:=SY<>COMMA;
              IF NOT TEST THEN INSYMBOL
            UNTIL TEST;
            INTEST(COLON,5);
            TYP(FSYS+[CASESY,SEMICOLON],LSP,LSIZE);
            WHILE NXT<>NXT1 DO
              BEGIN
                NXT↑.IDTYPE:=LSP; NXT↑.VADDR:=DISPL;
                NXT:=NXT↑.NEXT; DISPL:=DISPL+LSIZE
              END;
            NXT1:=LCP;
            IF SY=SEMICOLON THEN
              BEGIN INSYMBOL;
                TEST2([IDENT,CASESY],19,FSYS)
              END
          END (* WHILE *);
```

```
      NXT:=NIL;
      WHILE NXT1<>NIL DO
        BEGIN
          LCP:=NXT1↑.NEXT; NXT1↑.NEXT:=NXT; NXT:=NXT1; NXT1:=LCP
        END;
      IF SY=CASESY THEN
        BEGIN INSYMBOL;
          IF SY=IDENT THEN
            BEGIN PRTERR:=FALSE; SEARCHID([TYPES],LCP1);
              PRTERR:=TRUE;
              IF LCP1=NIL THEN
                BEGIN NEW(LCP); LCP↑.NAME:=ID; LCP↑.IDTYPE:=NIL;
                  LCP↑.KLASS:=FIELD; LCP↑.NEXT:=NIL;
                  LCP↑.VADDR:=DISPL;
                  ENTERID(LCP); INSYMBOL;
                  INTEST(COLON,5);
                  IF SY=IDENT THEN
                    BEGIN SEARCHID([TYPES],LCP1);
                      LSP1:=LCP1↑.IDTYPE;
                      IF LSP1<>NIL THEN
                        BEGIN DISPL:=DISPL+LSP1↑.SIZE;
                          IF (LSP1↑.FORM<=SUBRANGE) OR STRING(LSP1) THEN
                            BEGIN
                            (*IF LSP1=REALPTR THEN ERROR(109)
                              ELSE*)IF STRING(LSP1) THEN ERROR(399);
                              LCP↑.IDTYPE:=LSP1
                            END
                          ELSE ERROR(110)
                        END;
                      INSYMBOL
                    END
                  ELSE BEGIN ERROR(2); SKIP(FSYS+[COMMA,LPARENT]) END
                END
              ELSE BEGIN INSYMBOL; LSP1:=LCP1↑.IDTYPE END
            END
          ELSE BEGIN ERROR(2); SKIP(FSYS+[OFSY,LPARENT]) END;
          INTEST(OFSY,8);
          MINSIZE:=DISPL; MAXSIZE:=DISPL;
          REPEAT LSP2:=NIL;
            REPEAT CONSTANT(FSYS+[COMMA,COLON,LPARENT],LSP3,LVALU);
              IF NOT COMPTYPES(LSP1,LSP3) THEN ERROR(111);
              TEST:=SY<>COMMA;
              IF NOT TEST THEN INSYMBOL
            UNTIL TEST;
            INTEST(COLON,5); INTEST(LPARENT,9);
            FIELDLIST(FSYS+[RPARENT,SEMICOLON]);
            IF DISPL>MAXSIZE THEN MAXSIZE:=DISPL;
            IF SY=RPARENT THEN
              BEGIN INSYMBOL;
                TEST1(FSYS+[SEMICOLON],6)
              END
            ELSE ERROR(4);
            TEST:=SY<>SEMICOLON;
            IF NOT TEST THEN
              BEGIN DISPL:=MINSIZE; INSYMBOL END
          UNTIL TEST;
          DISPL:=MAXSIZE
        END
    END (* FIELDLIST *);

BEGIN (* TYP *)
  TEST2(TYPEBEGSYS,10,FSYS);
```

```
  IF SY IN TYPEBEGSYS THEN
    BEGIN
      IF SY IN SIMPTYPEBEGSYS THEN SIMPLETYPE(FSYS,FSP,FSIZE)
      ELSE
(* ↑ *) IF SY=ARROW THEN
           BEGIN NEW(LSP); FSP:=LSP;
             LSP↑.STYPE:=NIL; LSP↑.SIZE:=PTRSIZE; LSP↑.FORM:=POINTER;
             INSYMBOL;
             IF SY=IDENT THEN
               BEGIN PRTERR:=FALSE; (* NO ERROR IF NOT FOUND *)
                 SEARCHID([TYPES],LCP); PRTERR:=TRUE;
                 IF LCP=NIL THEN (* FORWARD REF. TYPE ID *)
                   BEGIN NEW(LCP);
                     LCP↑.NAME:=ID; LCP↑.IDTYPE:=LSP;
                     LCP↑.NEXT:=FWPTR; LCP↑.KLASS:=TYPES;
                     FWPTR:=LCP
                   END
                 ELSE
                   BEGIN
                     IF LCP↑.IDTYPE<>NIL THEN
                       LSP↑.STYPE:=LCP↑.IDTYPE
                   END;
                 INSYMBOL;
               END
             ELSE ERROR(2);
           END
         ELSE
           BEGIN
             IF SY=PACKEDSY THEN
               BEGIN INSYMBOL;
                 TEST2(TYPEDELS,10,FSYS)
               END;
             CASE SY OF
(* ARRAY *)   ARRAYSY:
                 BEGIN INSYMBOL;
                   INTEST(LBRACK,11);
                   LSP1:=NIL;
                   REPEAT NEW(LSP);
                     LSP↑.STYPE:=LSP1; LSP↑.INXTYPE:=NIL;
                     LSP↑.FORM:=ARRAYS;
                     LSP1:=LSP;
                     SIMPLETYPE(FSYS+[COMMA,RBRACK,OFSY],LSP2,LSIZE);
                     LSP1↑.SIZE:=LSIZE;
                     IF LSP2<>NIL THEN
                       IF LSP2↑.FORM<=SUBRANGE THEN
                         BEGIN
                           IF LSP2=INTPTR THEN
                             BEGIN ERROR(149); LSP2:=NIL END;
                           LSP↑.INXTYPE:=LSP2
                         END
                       ELSE BEGIN ERROR(113); LSP2:=NIL END;
                     TEST:=SY<>COMMA;
                     IF NOT TEST THEN INSYMBOL
                   UNTIL TEST;
                   INTEST(RBRACK,12); INTEST(OFSY,8);
                   TYP(FSYS,LSP,LSIZE);
                   REPEAT
                     LSP2:=LSP1↑.STYPE; LSP1↑.STYPE:=LSP;
                     IF LSP1↑.INXTYPE<>NIL THEN
                       BEGIN GETBOUNDS(LSP1↑.INXTYPE,LMIN,LMAX);
                         LSIZE:=LSIZE*(LMAX-LMIN+1);
                         LSP1↑.SIZE:=LSIZE
```

```
                    END;
                  LSP:=LSP1; LSP1:=LSP2
                UNTIL LSP1=NIL
              END;
(* RECORD *)  RECORDSY:
              BEGIN INSYMBOL;
                OLDTOP:=TOP;
                IF TOP<DISPLIMIT THEN
                  BEGIN TOP:=TOP+1;
                    DISPLAY[TOP]:=NIL
                  END
                ELSE ERROR(250);
                DISPL:=0;
                FIELDLIST(FSYS-[SEMICOLON]+[ENDSY]);
                NEW(LSP);
                LSP↑.FSTFLD:=DISPLAY[TOP];
                LSP↑.SIZE:=DISPL;
                LSP↑.FORM:=RECORDS;
                TOP:=OLDTOP;
                INTEST(ENDSY,13)
              END;
(* SET *)     SETSY:
              BEGIN INSYMBOL;
                INTEST(OFSY,8);
                SIMPLETYPE(FSYS,LSP1,LSIZE);
                IF LSP1<>NIL THEN
                  IF LSP1↑.FORM>SUBRANGE THEN
                    BEGIN ERROR(115); LSP1:=NIL END;
                NEW(LSP);
                LSP↑.STYPE:=LSP1; LSP↑.SIZE:=SETSIZE;
                LSP↑.FORM:=POWER
              END
          END;
          FSP:=LSP
        END;
      TEST1(FSYS,6)
    END
  ELSE FSP:=NIL;
  IF FSP=NIL THEN FSIZE:=1 ELSE FSIZE:=FSP↑.SIZE
END (* TYP *);

PROCEDURE CONSTDECLARATION;
VAR LCP: CTP; LSP: STP; LVALU: VALU;
BEGIN
  TEST2([IDENT],2,FSYS);
  WHILE SY=IDENT DO
    BEGIN NEW(LCP);
      LCP↑.NAME:=ID; LCP↑.IDTYPE:=NIL; LCP↑.NEXT:=NIL;
      LCP↑.KLASS:=KONST;
      INSYMBOL;
      IF (SY=RELOP) AND (OP=EQOP) THEN INSYMBOL ELSE ERROR(16);
      CONSTANT(FSYS+[SEMICOLON],LSP,LVALU);
      ENTERID(LCP);
      LCP↑.IDTYPE:=LSP; LCP↑.VALUES:=LVALU;
      IF SY=SEMICOLON THEN
        BEGIN INSYMBOL;
          TEST1(FSYS+[IDENT],6)
        END
      ELSE ERROR(14)
    END
END (* CONSTDECLARATION *);
```

```
PROCEDURE TYPEDECLARATION;
VAR LCP,LCP1,LCP2: CTP; LSP: STP; LSIZE: ADDRRANGE;
BEGIN
  TEST2([IDENT],2,FSYS);
  WHILE SY=IDENT DO
    BEGIN NEW(LCP);
      LCP↑.NAME:=ID; LCP↑.IDTYPE:=NIL; LCP↑.KLASS:=TYPES;
      INSYMBOL;
      IF (SY=RELOP) AND (OP=EQOP) THEN INSYMBOL ELSE ERROR(16);
      TYP(FSYS+[SEMICOLON],LSP,LSIZE);
      ENTERID(LCP);
      LCP↑.IDTYPE:=LSP;
      (* HAS ANY FORWARD REF. BEEN SATISFIED *)
      LCP1:=FWPTR;
      WHILE LCP1<>NIL DO
        BEGIN
          IF LCP1↑.NAME=LCP↑.NAME THEN
            BEGIN LCP1↑.IDTYPE↑.STYPE:=LCP↑.IDTYPE;
              IF LCP1<>FWPTR THEN
                LCP2↑.NEXT:=LCP1↑.NEXT
              ELSE FWPTR:=LCP1↑.NEXT;
            END;
          LCP2:=LCP1; LCP1:=LCP1↑.NEXT
        END;
      IF SY=SEMICOLON THEN
        BEGIN INSYMBOL;
          TEST1(FSYS+[IDENT],6)
        END
      ELSE ERROR(14)
    END;
  IF FWPTR<>NIL THEN
    BEGIN ERROR(117); WRITELN;
      REPEAT WRITELN(' TYPE-ID ',FWPTR↑.NAME);
        FWPTR:=FWPTR↑.NEXT
      UNTIL FWPTR=NIL;
    END
END (* TYPEDECLARATION *);

PROCEDURE VARDECLARATION;
VAR LCP,NXT: CTP; LSP: STP; LSIZE: ADDRRANGE;
BEGIN NXT:=NIL;
  REPEAT
  REPEAT
    IF SY=IDENT THEN
      BEGIN NEW(LCP);
        LCP↑.NAME:=ID; LCP↑.NEXT:=NXT; LCP↑.KLASS:=VARS;
        LCP↑.IDTYPE:=NIL; LCP↑.VKIND:=ACTUAL; LCP↑.VLEV:=LEVEL;
        ENTERID(LCP);
        NXT:=LCP;
        INSYMBOL
      END
    ELSE ERROR(2);
    TEST2(FSYS+TYPEDELS+[COMMA,COLON],6,[SEMICOLON]);
    TEST:=SY<>COMMA;
    IF NOT TEST THEN INSYMBOL;
  UNTIL TEST;
  INTEST(COLON,5);
  TYP(FSYS+[SEMICOLON]+TYPEDELS,LSP,LSIZE);
  WHILE NXT<>NIL DO
    BEGIN NXT↑.IDTYPE:=LSP; LC:=LC+LSIZE;
      NXT↑.VADDR:=LC; NXT:=NXT↑.NEXT
    END;
```

```
    IF SY=SEMICOLON THEN
      BEGIN INSYMBOL;
        TEST1(FSYS+[IDENT],6)
      END
    ELSE ERROR(14)
    UNTIL (SY<>IDENT) AND NOT(SY IN TYPEDELS);
    IF FWPTR<>NIL THEN
      BEGIN ERROR(118); WRITELN;
        REPEAT WRITELN(' TYPE-ID ',FWPTR↑.NAME);
          FWPTR:=FWPTR↑.NEXT
        UNTIL FWPTR=NIL;
      END
  END (* VARDECLARATION *);

  PROCEDURE PROCDECLARATION(FSY: SYMBOL);
  VAR OLDLEV: 0..MAXLEVEL; LSY: SYMBOL; LCP,LCP1: CTP; LSP: STP;
      FORW: BOOLEAN; OLDTOP: DISPRANGE;
      LLC,LCM: ADDRRANGE; MARKP: ↑CHAR;

    PROCEDURE PARAMETERLIST(FSY: SETOFSYS; VAR FPAR: CTP);
    VAR LCP,LCP1,LCP2,LCP3: CTP; LSP: STP; LKIND: IDKIND;
        LLC,LEN: ADDRRANGE; COUNT,OFFSET: INTEGER;
    BEGIN LCP1:=NIL;
      TEST1(FSY+[LPARENT],7);
      IF SY=LPARENT THEN
        BEGIN IF FORW THEN ERROR(119);
          INSYMBOL;
          IF NOT(SY IN [IDENT,VARSY]) THEN
            BEGIN ERROR(7); SKIP(FSYS+[IDENT,RPARENT]) END;
          WHILE SY IN [IDENT,VARSY] DO
            BEGIN
              IF SY=VARSY THEN
                BEGIN LKIND:=FORMAL; INSYMBOL END
              ELSE LKIND:=ACTUAL;
              LCP2:=NIL;
              COUNT:=0;
              REPEAT
                IF SY=IDENT THEN
                  BEGIN NEW(LCP);
                    LCP↑.NAME:=ID; LCP↑.IDTYPE:=NIL;
                    LCP↑.KLASS:=VARS; LCP↑.VKIND:=LKIND;
                    LCP↑.NEXT:=LCP2; LCP↑.VLEV:=LEVEL;
                    ENTERID(LCP);
                    LCP2:=LCP; COUNT:=COUNT+1;
                    INSYMBOL;
                  END;
                IF NOT(SY IN [COMMA,COLON]+FSYS) THEN
                  BEGIN ERROR(7);
                    SKIP(FSYS+[COMMA,SEMICOLON,RPARENT])
                  END;
                TEST:=SY<>COMMA;
                IF NOT TEST THEN INSYMBOL
              UNTIL TEST;
              IF SY=COLON THEN
                BEGIN INSYMBOL;
                  IF SY=IDENT THEN
                    BEGIN SEARCHID([TYPES],LCP);
                      LSP:=LCP↑.IDTYPE;
                      LCP3:=LCP2;
                      OFFSET:=0; LEN:=PTRSIZE;
                      IF (LKIND=ACTUAL) AND (LSP↑.FORM<ARRAYS)
                      THEN
```

```
                    IF LSP↑.SIZE=1 THEN OFFSET:=1
                    ELSE LEN:=LSP↑.SIZE;
                    LC:=LC+COUNT*LEN;
                    LLC:=LC;
                    WHILE LCP2<>NIL DO
                      BEGIN LCP:=LCP2;
                        LCP2↑.IDTYPE:=LSP; LCP2↑.VADDR:=LLC-OFFSET;
                        LLC:=LLC-LEN; LCP2:=LCP2↑.NEXT
                      END;
                    LCP↑.NEXT:=LCP1; LCP1:=LCP3;
                    INSYMBOL
                  END
                ELSE ERROR(2);
                TEST1(FSYS+[SEMICOLON,RPARENT],7)
              END
            ELSE ERROR(5);
            IF SY=SEMICOLON THEN
              BEGIN INSYMBOL;
                IF NOT(SY IN FSYS+[IDENT,VARSY]) THEN
                  BEGIN ERROR(7); SKIP(FSYS+[IDENT,RPARENT]) END
              END
          END (* WHILE *);
        IF SY=RPARENT THEN
          BEGIN INSYMBOL;
            TEST1(FSY+FSYS,6)
          END
        ELSE ERROR(4);
        LCP3:=NIL;
        (* REVERSE POINTERS AND RESERVE LOCAL CELLS FOR COPIES OF
           MULTIPLE VALUES *)
        WHILE LCP1<>NIL DO
          BEGIN LCP2:=LCP1↑.NEXT; LCP1↑.NEXT:=LCP3;
            IF LCP1↑.KLASS=VARS THEN
              IF LCP1↑.IDTYPE<>NIL THEN
                IF (LCP1↑.VKIND=ACTUAL)AND(LCP1↑.IDTYPE↑.FORM>POWER)
                THEN
                  BEGIN LC:=LC+LCP1↑.IDTYPE↑.SIZE; LCP1↑.VADDR:=LC
                  END;
            LCP3:=LCP1; LCP1:=LCP2
          END;
        FPAR:=LCP3
      END
    ELSE FPAR:=NIL
  END (* PARAMETERLIST *);

BEGIN (* PROCDECLARATION *)
  LLC:=LC; LC:=LCAFTERMARKSTACK;
  IF SY=IDENT THEN
    BEGIN SEARCHSECTION(DISPLAY[TOP],LCP); (* DECIDE IF FORW *)
      IF LCP<>NIL THEN
        BEGIN
          IF LCP↑.KLASS=PROC THEN
            FORW:=LCP↑.FORWDECL AND(FSY=PROCSY)
          ELSE
            IF LCP↑.KLASS=FUNC THEN
              FORW:=LCP↑.FORWDECL AND(FSY=FUNCSY)
            ELSE FORW:=FALSE;
          IF NOT FORW THEN ERROR(160)
        END
      ELSE FORW:=FALSE;
      IF NOT FORW THEN
        BEGIN
```

```
1117            NEW(LCP);
1118            LCP↑.NAME:=ID; LCP↑.IDTYPE:=NIL;
1119            LCP↑.PFLEV:=LEVEL; LCP↑.PFNAME:=NPROC; NPROC:=NPROC+1;
1120            LCP↑.EXTERNL:=FALSE;
1121            LCP↑.PFDECKIND:=DECLARED;
1122            IF FSY=PROCSY THEN LCP↑.KLASS:=PROC
1123            ELSE LCP↑.KLASS:=FUNC;
1124            ENTERID(LCP)
1125          END
1126        ELSE
1127          BEGIN LCP1:=LCP↑.NEXT;
1128            WHILE LCP1<>NIL DO
1129              BEGIN
1130                IF LCP1↑.KLASS=VARS THEN
1131                  IF LCP1↑.IDTYPE<>NIL THEN
1132                    BEGIN LCM:=LCP1↑.VADDR;
1133                      IF LCM>LC THEN LC:=LCM
1134                    END;
1135                LCP1:=LCP1↑.NEXT
1136              END
1137          END;
1138        INSYMBOL
1139      END
1140    ELSE ERROR(2);
1141    OLDLEV:=LEVEL; OLDTOP:=TOP;
1142    IF LEVEL<MAXLEVEL THEN LEVEL:=LEVEL+1 ELSE ERROR(251);
1143    IF TOP<DISPLIMIT THEN
1144      BEGIN TOP:=TOP+1; SAVETOP:=TOP;
1145        IF FORW THEN DISPLAY[TOP]:=LCP↑.NEXT
1146        ELSE DISPLAY[TOP]:=NIL
1147      END
1148    ELSE ERROR(250);
1149    IF FSY=PROCSY THEN
1150      BEGIN PARAMETERLIST([SEMICOLON],LCP1);
1151        IF NOT FORW THEN LCP↑.NEXT:=LCP1
1152      END
1153    ELSE
1154      BEGIN PARAMETERLIST([SEMICOLON,COLON],LCP1);
1155        IF NOT FORW THEN LCP↑.NEXT:=LCP1;
1156        IF SY=COLON THEN
1157          BEGIN INSYMBOL;
1158            IF SY=IDENT THEN
1159              BEGIN IF FORW THEN ERROR(122);
1160                SEARCHID([TYPES],LCP1);
1161                LSP:=LCP1↑.IDTYPE;
1162                LCP↑.IDTYPE:=LSP;
1163                IF LSP<>NIL THEN
1164                  IF LSP↑.FORM>POWER THEN
1165                    BEGIN ERROR(120); LCP↑.IDTYPE:=NIL END;
1166                INSYMBOL
1167              END
1168            ELSE BEGIN ERROR(2); SKIP(FSYS+[SEMICOLON]) END
1169          END
1170        ELSE
1171          IF NOT FORW THEN ERROR(123)
1172      END;
1173    INTEST(SEMICOLON,14);
1174    IF SY=FORWARDSY THEN
1175      BEGIN
1176        IF FORW THEN ERROR(161)
1177        ELSE
1178          LCP↑.FORWDECL:=TRUE;
```

```
1179        INSYMBOL;
1180        INTEST(SEMICOLON,14);
1181        TEST1(FSYS,6)
1182      END
1183    ELSE
1184      IF SY=EXTERNSY THEN
1185        BEGIN
1186          IF FORW THEN ERROR(161);
1187          INSYMBOL; LCP↑.EXTERNL:=TRUE;
1188          IF (SY=RELOP) AND (OP=EQOP) THEN
1189            BEGIN INSYMBOL;
1190              IF SY=INTCONST THEN LCP↑.PFNAME:=VAL.IVAL
1191              ELSE ERROR(106)
1192            END
1193          ELSE ERROR(16);
1194          REPEAT INSYMBOL UNTIL SY=SEMICOLON; INSYMBOL;
1195          TEST1(FSYS,6)
1196        END
1197      ELSE
1198        BEGIN LCP↑.FORWDECL:=FALSE; NEW(MARKP);
1199          REPEAT BLOCK(FSYS,SEMICOLON,LCP);
1200            IF SY=SEMICOLON THEN
1201              BEGIN INSYMBOL;
1202                IF NOT(SY IN [BEGINSY,PROCSY,FUNCSY]) THEN
1203                  BEGIN ERROR(6); SKIP(FSYS) END
1204              END
1205            ELSE ERROR(14)
1206          UNTIL SY IN [BEGINSY,PROCSY,FUNCSY];
1207          RELEASE(MARKP)
1208        END;
1209    LEVEL:=OLDLEV; TOP:=OLDTOP; LC:=LLC;
1210  END (* PROCDECLARATION *);
1211
1212  PROCEDURE BODY(FSYS: SETOFSYS);
1213  VAR I,ENTNAME,SEGSIZE: INTEGER;
1214      LCMAX,LLC1: ADDRRANGE; LCP: CTP;
1215
1216    PROCEDURE LDAGEN(LEV: INTEGER; DPLMT: ADDRRANGE);
1217    BEGIN
1218      IF DPLMT<256 THEN
1219        BEGIN BYTEGEN(16(*LDAS*)+LEV); BYTEGEN(DPLMT) END
1220      ELSE
1221        BEGIN BYTEGEN(32(*LDA*)+LEV); ADDRGEN(DPLMT) END
1222    END (* LDAGEN *);
1223
1224    PROCEDURE LODGEN(LEV: INTEGER; VAR FATTR: ATTR);
1225    BEGIN
1226      IF (FATTR.TYPTR↑.SIZE<=2)AND(FATTR.DPLMT<256) THEN
1227        BEGIN BYTEGEN(64+16*GATTR.TYPTR↑.SIZE+LEV); BYTEGEN(FATTR.DPLMT)
1228        END
1229      ELSE
1230        BEGIN LDAGEN(LEV,FATTR.DPLMT);
1231          CASE FATTR.TYPTR↑.SIZE OF
1232            1: BYTEGEN(154);
1233            2: BYTEGEN(155);
1234            8: BYTEGEN(156)
1235          END
1236        END
1237    END (* LODGEN *);
1238
1239    PROCEDURE CONDGEN(FOP: OPRANGE; VAR FATTR: ATTR);
1240    BEGIN
```

```
1241    IF FATTR.TYPTR↑.FORM>POWER THEN
1242      BEGIN BYTEGEN(FOP+2); ADDRGEN(FATTR.TYPTR↑.SIZE) END
1243    ELSE
1244      IF FATTR.TYPTR↑.SIZE=1 THEN BYTEGEN(FOP)
1245      ELSE
1246        IF FATTR.TYPTR↑.SIZE=2 THEN BYTEGEN(FOP)
1247        ELSE
1248          IF FATTR.TYPTR↑.SIZE=8 THEN BYTEGEN(FOP+3)
1249          ELSE ERROR(401)  (* WRONG SIZE  *)
1250  END (* CONDGEN *);
1251
1252  PROCEDURE LDCIGEN(VAL: INTEGER);
1253  BEGIN
1254    IF (VAL<16) AND (VAL>=0) THEN BYTEGEN(VAL)
1255    ELSE
1256      IF VAL<0 THEN
1257        BEGIN BYTEGEN(176(*LNC*)); ADDRGEN(-VAL) END
1258      ELSE
1259        BEGIN BYTEGEN(160(*LDC*)); ADDRGEN(VAL) END
1260  END (* LDCIGEN *);
1261
1262  PROCEDURE LDCS(S: INTSET);
1263  VAR I,K,L,N: INTEGER; B: BOOLEAN;
1264  BEGIN N:=0; B:=FALSE;
1265    FOR I:=0 TO SETMAX DO
1266      IF I IN S THEN N:=N+1;
1267    IF N=0 THEN BYTEGEN(193(*LNS*)) (* LOAD EMPTY SET *)
1268    ELSE
1269      IF N<4 THEN
1270        FOR I:=0 TO SETMAX DO
1271          IF I IN S THEN
1272            BEGIN LDCIGEN(I); BYTEGEN(174(*SGS*));
1273              IF B THEN BYTEGEN(175(*UNI*));
1274              B:=TRUE
1275            END
1276          ELSE
1277      ELSE
1278        BEGIN BYTEGEN(186(*LDCS*)); L:=0; K:=128;
1279          FOR I:=0 TO SETMAX DO
1280            BEGIN
1281              IF K<1 THEN BEGIN K:=128; BYTEGEN(L); L:=0 END;
1282              IF I IN S THEN L:=L+K;
1283              K:=K DIV 2
1284            END;
1285          BYTEGEN(L)
1286        END
1287  END (* LDCS *);
1288
1289  PROCEDURE CSPGEN(P: INTEGER);
1290  BEGIN
1291    BYTEGEN(189(*CSP*)); BYTEGEN(P)
1292  END (* CSPGEN *);
1293
1294  PROCEDURE GENINC(INC: INTEGER);
1295  BEGIN
1296    IF INC<>0 THEN
1297      IF INC=1 THEN BYTEGEN(185(*INC1*))
1298      ELSE
1299        IF INC=-1 THEN BYTEGEN(184(*DEC1*))
1300        ELSE
1301          IF INC>0 THEN
1302            BEGIN BYTEGEN(180(*INC*)); ADDRGEN(INC) END
```

```
1303            ELSE
1304              BEGIN BYTEGEN(179(*DEC*)); ADDRGEN(-INC) END
1305    END (* GENINC *);
1306
1307    PROCEDURE LOAD;
1308    BEGIN
1309      IF GATTR.TYPTR<>NIL THEN
1310        BEGIN
1311          IF GATTR.KIND=CST THEN
1312            IF (GATTR.TYPTR↑.FORM=SCALAR) THEN LDCIGEN(GATTR.CVAL.IVAL)
1313            ELSE
1314            IF GATTR.TYPTR=NILPTR THEN BYTEGEN(0)
1315              ELSE LDCS(GATTR.CVAL.VALP↑.PVAL)
1316          ELSE
1317            IF GATTR.KIND=VARBL THEN
1318              IF GATTR.ACCESS=DRCT THEN
1319                LODGEN(LEVEL-GATTR.LEVEL,GATTR)
1320              ELSE
1321                BEGIN
1322                  GENINC(GATTR.DPLMT);
1323                  CASE GATTR.TYPTR↑.SIZE OF
1324                    1: BYTEGEN(154);
1325                    2: BYTEGEN(155);
1326                    8: BYTEGEN(156)
1327                  END;
1328                END;
1329          GATTR.KIND:=EXPR
1330        END
1331    END (* LOAD *);
1332
1333    PROCEDURE STORE(VAR FATTR: ATTR);
1334    BEGIN
1335      IF FATTR.TYPTR<>NIL THEN
1336        IF FATTR.ACCESS=DRCT THEN
1337          BEGIN BYTEGEN(96+16*FATTR.TYPTR↑.SIZE+LEVEL-FATTR.LEVEL);
1338            BYTEGEN(FATTR.DPLMT)
1339          END
1340        ELSE
1341          IF FATTR.DPLMT<>0 THEN ERROR(400)
1342          ELSE
1343            CASE FATTR.TYPTR↑.SIZE OF
1344              1: BYTEGEN(157);
1345              2: BYTEGEN(158);
1346              8: BYTEGEN(159)
1347            END
1348    END (* STORE *);
1349
1350    PROCEDURE LOADADDRESS;
1351    VAR LVP: CSP; I: INTEGER;
1352    BEGIN
1353      IF GATTR.TYPTR<>NIL THEN
1354        BEGIN
1355          IF GATTR.KIND=CST THEN
1356            IF STRING(GATTR.TYPTR) THEN
1357              BEGIN BYTEGEN(188(*LCA*));
1358                BYTEGEN(STRINGSIZE(GATTR.CVAL.VALP));
1359                LVP:=GATTR.CVAL.VALP;
1360                REPEAT
1361                  FOR I:=1 TO LVP↑.SLGTH DO
1362                    BYTEGEN(LVP↑.SVAL[I]);
1363                  LVP:=LVP↑.SPTR
1364                UNTIL LVP=NIL
```

```
1365              END
1366            ELSE ERROR(400)
1367          ELSE
1368            IF GATTR.KIND=VARBL THEN
1369              IF GATTR.ACCESS=DRCT THEN
1370                LDAGEN(LEVEL-GATTR.LEVEL,GATTR.DPLMT)
1371              ELSE GENINC(GATTR.DPLMT)
1372            ELSE ERROR(400);
1373          GATTR.KIND:=VARBL; GATTR.ACCESS:=INDRCT; GATTR.DPLMT:=0
1374        END
1375    END (* LOADADDRESS *);
1376
1377    PROCEDURE GENFJP(FADDR: INTEGER);
1378    BEGIN LOAD;
1379      IF GATTR.TYPTR<>NIL THEN
1380        IF GATTR.TYPTR<>BOOLPTR THEN ERROR(144);
1381      BYTEGEN(177(*FJP*)); ADDRGEN(FADDR)
1382    END (* GENFJP *);
1383
1384    PROCEDURE GENCUP(FP1,FP2: INTEGER);
1385    BEGIN
1386      IF PFCTTEST THEN
1387        BEGIN BYTEGEN(FP1); BYTEGEN(191(*CUP2*)) END
1388      ELSE BYTEGEN(190(*CUP1*));
1389      BYTEGEN(FP2)
1390    END (* GENCUP *);
1391
1392    PROCEDURE STATEMENT(FSYS: SETOFSYS);
1393    VAR LCP: CTP;
1394
1395      PROCEDURE EXPRESSION(FSYS: SETOFSYS); FORWARD;
1396
1397      PROCEDURE SELECTOR(FSYS: SETOFSYS; FCP: CTP);
1398      VAR LATTR: ATTR; LCP: CTP; LMIN,LMAX: INTEGER;
1399      BEGIN
1400        GATTR.TYPTR:=FCP↑.IDTYPE; GATTR.KIND:=VARBL;
1401        IF FCP↑.KLASS=VARS THEN
1402          IF GATTR.TYPTR<>NIL THEN
1403            IF (FCP↑.VKIND=ACTUAL) AND (FCP↑.VADDR<256) AND
1404              (GATTR.TYPTR↑.SIZE<=2) THEN
1405              BEGIN GATTR.ACCESS:=DRCT; GATTR.LEVEL:=FCP↑.VLEV;
1406                GATTR.DPLMT:=FCP↑.VADDR
1407              END
1408            ELSE
1409              BEGIN GATTR.ACCESS:=INDRCT; GATTR.DPLMT:=0;
1410                IF FCP↑.VKIND=FORMAL THEN
1411                  BEGIN BYTEGEN(96+LEVEL-FCP↑.VLEV);
1412                    BYTEGEN(FCP↑.VADDR)
1413                  END
1414                ELSE LDAGEN(LEVEL-FCP↑.VLEV,FCP↑.VADDR)
1415              END
1416          ELSE
1417        ELSE
1418          IF FCP↑.KLASS=FUNC THEN
1419            IF FCP↑.PFDECKIND=STANDARD THEN ERROR(150)
1420            ELSE
1421              BEGIN GATTR.ACCESS:=DRCT; GATTR.LEVEL:=FCP↑.PFLEV+1;
1422                GATTR.DPLMT:=0 (* IMPL. RELAT. ADDR. OF FCT. RESULT *)
1423              END;
1424        TEST1(SELECTSYS+FSYS,59);
1425        WHILE SY IN SELECTSYS DO
1426          BEGIN
```

```
1427 (* [ *)    IF SY=LBRACK THEN
1428               BEGIN
1429                 REPEAT LATTR:=GATTR;
1430                   IF LATTR.TYPTR<>NIL THEN
1431                     IF LATTR.TYPTR↑.FORM<>ARRAYS THEN
1432                       BEGIN ERROR(138); LATTR.TYPTR:=NIL END;
1433                   LOADADDRESS;
1434                   INSYMBOL; EXPRESSION(FSYS+[COMMA,RBRACK]);
1435                   IF GATTR.KIND<>CST THEN LOAD;
1436                   IF GATTR.TYPTR<>NIL THEN
1437                     IF GATTR.TYPTR↑.FORM>SUBRANGE THEN ERROR(113);
1438                   IF LATTR.TYPTR<>NIL THEN
1439                     BEGIN
1440                       IF COMPTYPES(LATTR.TYPTR↑.INXTYPE,GATTR.TYPTR) THE
1441                         BEGIN
1442                           IF LATTR.TYPTR↑.INXTYPE<>NIL THEN
1443                             BEGIN
1444                               GETBOUNDS(LATTR.TYPTR↑.INXTYPE,LMIN,LMAX);
1445                               IF GATTR.KIND=CST THEN
1446                                 BEGIN
1447                                   IF (GATTR.CVAL.IVAL<LMIN) OR
1448                                     (GATTR.CVAL.IVAL>LMAX) THEN
1449                                     ERROR(302);
1450                                   GATTR.CVAL.IVAL:=GATTR.CVAL.IVAL-LMIN
1451                                 END
1452                               ELSE GENINC(-LMIN)
1453                             END
1454                         END
1455                       ELSE ERROR(139);
1456                       GATTR.TYPTR:=LATTR.TYPTR↑.STYPE;
1457                       IF GATTR.TYPTR<>NIL THEN
1458                         IF GATTR.KIND=CST THEN
1459                           BEGIN GATTR.CVAL.IVAL:=GATTR.CVAL.IVAL*
1460                               GATTR.TYPTR↑.SIZE; LDCIGEN(GATTR.CVAL.IVAL
1461                           END
1462                         ELSE
1463                           BEGIN
1464                             IF GATTR.TYPTR↑.SIZE<>1 THEN
1465                               BEGIN LDCIGEN(GATTR.TYPTR↑.SIZE);
1466                                 BYTEGEN(170(*MPI*))
1467                               END
1468                           END;
1469                       BYTEGEN(162(*ADI*));
1470                       GATTR.KIND:=VARBL; GATTR.ACCESS:=INDRCT;
1471                       GATTR.DPLMT:=0
1472                     END
1473                   UNTIL SY<>COMMA;
1474                 INTEST(RBRACK,12)
1475               END (* IF SY=LBRACK *)
1476             ELSE
1477 (* . *)       IF SY=PERIOD THEN
1478                 BEGIN
1479                   IF GATTR.TYPTR<>NIL THEN
1480                     IF GATTR.TYPTR↑.FORM<>RECORDS THEN
1481                       BEGIN ERROR(140); GATTR.TYPTR:=NIL END;
1482                   INSYMBOL;
1483                   IF SY=IDENT THEN
1484                     BEGIN
1485                       IF GATTR.TYPTR<>NIL THEN
1486                         BEGIN SEARCHSECTION(GATTR.TYPTR↑.FSTFLD,LCP);
1487                           IF LCP=NIL THEN
1488                             BEGIN ERROR(152); GATTR.TYPTR:=NIL END
```

```
                        ELSE
                          BEGIN GATTR.TYPTR:=LCP↑.IDTYPE;
                            IF GATTR.ACCESS=INDRCT THEN
                              GATTR.DPLMT:=GATTR.DPLMT+LCP↑.VADDR
                            ELSE
                              GATTR.DPLMT:=GATTR.DPLMT-LCP↑.VADDR
                          END
                      END;
                    INSYMBOL
                  END (* SY=IDENT *)
                ELSE ERROR(2)
              END (* SY=PERIOD *)
            ELSE
(* ↑ *)        BEGIN
                IF GATTR.TYPTR<>NIL THEN
                  IF GATTR.TYPTR↑.FORM=POINTER THEN
                    BEGIN LOAD; GATTR.TYPTR:=GATTR.TYPTR↑.STYPE;
                      GATTR.KIND:=VARBL; GATTR.ACCESS:=INDRCT;
                      GATTR.DPLMT:=0
                    END
                  ELSE ERROR(141);
                INSYMBOL
              END;
          TEST1(FSYS+SELECTSYS,6)
        END (* WHILE *)
    END (* SELECTOR *);

    PROCEDURE CALL(FSYS: SETOFSYS; FCP: CTP);
    VAR LKEY: 1..8;

      PROCEDURE VARIABLE(FSYS: SETOFSYS);
      VAR LCP: CTP;
      BEGIN
        IF SY=IDENT THEN
          BEGIN SEARCHID([VARS],LCP); INSYMBOL END
        ELSE BEGIN ERROR(2); LCP:=UVARPTR END;
        SELECTOR(FSYS,LCP)
      END (* VARIABLE *);

    PROCEDURE READ;
    BEGIN
      IF SY=LPARENT THEN
        BEGIN INSYMBOL;
          IF SY=IDENT THEN
            BEGIN
              REPEAT VARIABLE(FSYS+[COMMA,RPARENT]); LOADADDRESS;
                IF GATTR.TYPTR<>NIL THEN
                  IF GATTR.TYPTR↑.FORM<=SUBRANGE THEN
                    IF COMPTYPES(INTPTR,GATTR.TYPTR) THEN
                      CSPGEN(3(*RDI*))
                    ELSE
                        IF COMPTYPES(CHARPTR,GATTR.TYPTR) THEN
                          CSPGEN(5(*RDC*))
                        ELSE ERROR(399)
                  ELSE ERROR(116);
                TEST:=SY<>COMMA;
                IF NOT TEST THEN INSYMBOL
              UNTIL TEST;
            END;
          INTEST(RPARENT,4)
        END;
        IF LKEY=5 THEN CSPGEN(4(*RLN*))
```

```
1561      END (* READ *);
1562
1563      PROCEDURE WRITE;
1564      VAR LSP: STP; DEFAULT: BOOLEAN; LLKEY: 1..8;
1565          LEN: ADDRRANGE; ;
1566      BEGIN LLKEY:=LKEY;
1567        IF SY=LPARENT THEN
1568          BEGIN INSYMBOL;
1569            IF SY IN FACBEGSYS THEN
1570              BEGIN
1571                REPEAT EXPRESSION(FSYS+[COMMA,COLON,RPARENT]);
1572                  LSP:=GATTR.TYPTR;
1573                  IF LSP<>NIL THEN
1574                    IF LSP↑.FORM<=SUBRANGE THEN LOAD ELSE LOADADDRESS;
1575                  IF SY=COLON THEN
1576                    BEGIN INSYMBOL; EXPRESSION(FSYS+[COMMA,RPARENT]);
1577                      IF GATTR.TYPTR<>NIL THEN
1578                        IF GATTR.TYPTR<>INTPTR THEN ERROR(116);
1579                      LOAD; DEFAULT:=FALSE
1580                    END
1581                  ELSE DEFAULT:=TRUE;
1582                    IF LSP=INTPTR THEN
1583                      BEGIN IF DEFAULT THEN LDCIGEN(6);
1584                        CSPGEN(0(*WRI*))
1585                      END
1586                    ELSE
1587                        IF LSP=CHARPTR THEN
1588                          BEGIN IF DEFAULT THEN LDCIGEN(1);
1589                            CSPGEN(1(*WRC*))
1590                          END
1591                        ELSE
1592                          IF LSP<>NIL THEN
1593                            BEGIN
1594                              IF LSP↑.FORM=SCALAR THEN ERROR(399)
1595                              ELSE
1596                                IF STRING(LSP) THEN
1597                                  BEGIN LEN:=LSP↑.SIZE;
1598                                    IF DEFAULT THEN LDCIGEN(LEN);
1599                                    LDCIGEN(LEN);
1600                                    CSPGEN(2(*WRS*))
1601                                  END
1602                                ELSE ERROR(116)
1603                            END;
1604                  TEST:=SY<>COMMA;
1605                IF NOT TEST THEN INSYMBOL
1606              UNTIL TEST;
1607            END;
1608        INTEST(RPARENT,4)
1609      END;
1610      IF LLKEY=6 THEN (*WRITELN*)
1611        CSPGEN(6(*WLN*))
1612    END (* WRITE *);
1613
1614    PROCEDURE NEW;
1615    VAR LSIZE: ADDRRANGE;
1616    BEGIN VARIABLE(FSYS+[COMMA,RPARENT]); LOADADDRESS;
1617      LSIZE:=0;
1618      IF GATTR.TYPTR<>NIL THEN
1619        IF GATTR.TYPTR↑.FORM=POINTER THEN
1620          BEGIN
1621            IF GATTR.TYPTR↑.STYPE<>NIL THEN
1622              LSIZE:=GATTR.TYPTR↑.STYPE↑.SIZE
```

```
1613            END
1614          ELSE ERROR(116);
1615        LOCIGEN(LSIZE);
1616        CSPGEN(7(*NEW*))
1617      END (* NEW *);
1618
1619      PROCEDURE RELEASE;
1620      BEGIN VARIABLE(FSYS+[RPARENT]);
1621        IF GATTR.TYPTR<>NIL THEN
1622          IF GATTR.TYPTR↑.FORM=POINTER THEN
1623            BEGIN LOAD; CSPGEN(9(*RST*)) END
1624          ELSE ERROR(125)
1625      END (* RELEASE *);
1626
1627      PROCEDURE ORD;
1628      BEGIN
1629        IF GATTR.TYPTR<>NIL THEN
1630          IF GATTR.TYPTR↑.FORM>=POWER THEN ERROR(125);
1631        GATTR.TYPTR:=INTPTR
1632      END (* ORD *);
1633
1634      PROCEDURE CHR;
1635      BEGIN
1636        IF GATTR.TYPTR<>NIL THEN
1637          IF GATTR.TYPTR<>INTPTR THEN ERROR(125);
1638        GATTR.TYPTR:=CHARPTR
1639      END (* CHR *);
1640
1641      PROCEDURE ODD;
1642      BEGIN
1643        IF GATTR.TYPTR<>NIL THEN
1644          IF GATTR.TYPTR<>INTPTR THEN ERROR(125);
1645        CSPGEN(12(*ODD*)); GATTR.TYPTR:=BOOLPTR
1646      END (* ODD *);
1647
1648      PROCEDURE EOF;
1649      BEGIN
1650        CSPGEN(8(*EOF*)); GATTR.TYPTR:=BOOLPTR
1651      END (* EOF *);
1652
1653      PROCEDURE EOLN;
1654      BEGIN
1655        CSPGEN(10(*ELN*));
1656        GATTR.TYPTR:=BOOLPTR
1657      END (* EOLN *);
1658
1659      PROCEDURE CALLNONSTANDARD;
1660       VAR NXT,LCP: CTP; LSP: STP;
1661          LOCPAR,LLC: ADDRRANGE;
1662      BEGIN LOCPAR:=0; NXT:=FCP↑.NEXT; PFCTTEST:=FALSE;
1663        IF NOT FCP↑.EXTERNL THEN
1664          IF FCP↑.KLASS=PROC THEN
1665            BYTEGEN(48(*MSTO*)+LEVEL-FCP↑.PFLEV)
1666          ELSE
1667            BEGIN BYTEGEN(64(*MSTN*)+LEVEL-FCP↑.PFLEV);
1668              BYTEGEN(FCP↑.IDTYPE↑.SIZE)
1669            END;
1670        IF SY=LPARENT THEN
1671          BEGIN LLC:=LC;
1672            REPEAT INSYMBOL;
1673              EXPRESSION(FSYS+[COMMA,RPARENT]);
1674              IF GATTR.TYPTR<>NIL THEN
```

```
1675                BEGIN
1676                  IF NXT<>NIL THEN
1677                    BEGIN LSP:=NXT↑.IDTYPE;
1678                      IF LSP<>NIL THEN
1679                        BEGIN
1680                          IF (NXT↑.VKIND=ACTUAL) THEN
1681                            IF LSP↑.FORM<ARRAYS THEN
1682                              BEGIN LOAD;
1683                                LOCPAR:=LOCPAR+LSP↑.SIZE
1684                              END
1685                            ELSE
1686                              BEGIN
1687                                IF (GATTR.KIND=EXPR)
1688                                  OR (GATTR.KIND=CST) THEN
1689                                  BEGIN LOAD;
1690                                    LODGEN(0,GATTR);
1691                                    LC:=LC+GATTR.TYPTR↑.SIZE;
1692                                    LOAGEN(0,LC);
1693                                    IF LCMAX<LC THEN LCMAX:=LC
1694                                  END
1695                                ELSE
1696                                  LOADADDRESS;
1697                                LOCPAR:=LOCPAR+PTRSIZE
1698                              END
1699                          ELSE
1700                            IF GATTR.KIND=VARBL THEN
1701                              BEGIN LOADADDRESS; LOCPAR:=LOCPAR+PTRSIZE
1702                              END
1703                            ELSE ERROR(154);
1704                          IF NOT COMPTYPES(LSP,GATTR.TYPTR) THEN
1705                            ERROR(142)
1706                        END
1707                    END;
1708                  IF NXT<>NIL THEN NXT:=NXT↑.NEXT
1709                UNTIL SY<>COMMA;
1710                LC:=LLC;
1711                INTEST(RPARENT,4)
1712              END (* IF SY=LPARENT *);
1713              IF NXT<>NIL THEN ERROR(126);
1714              IF FCP↑.EXTERNL THEN
1715                BEGIN BYTEGEN(LOCPAR); BYTEGEN(187(*CAP*));
1716                  ADDRGEN(FCP↑.PFNAME)
1717                END
1718              ELSE GENCUP(LOCPAR,FCP↑.PFNAME);
1719              IF FCP↑.IDTYPE<>NIL THEN
1720                IF FCP↑.IDTYPE↑.SIZE=1 THEN BYTEGEN(192(*FIX21*));
1721              PFCTTEST:=TRUE;
1722              GATTR.TYPTR:=FCP↑.IDTYPE
1723            END (* CALLNONSTANDARD *);
1724
1725        BEGIN (* CALL *)
1726          IF FCP↑.PFDECKIND=STANDARD THEN
1727            BEGIN
1728              LKEY:=FCP↑.KEY;
1729              IF FCP↑.KLASS=PROC THEN
1730                IF LKEY=8 THEN BYTEGEN(161(*RETP*))
1731                ELSE
1732                IF (LKEY=1) OR (LKEY=5) THEN READ
1733                ELSE
1734                  IF (LKEY=2) OR (LKEY=6) THEN WRITE
1735                  ELSE
```

```
                BEGIN
                  INTEST(LPARENT,9);
                  IF LKEY=3 THEN NEW
                  ELSE
                    IF LKEY=4 THEN RELEASE
                    ELSE ERROR(400);
                  INTEST(RPARENT,4)
                END
            ELSE
              IF LKEY<4 THEN
                BEGIN
                  INTEST(LPARENT,9);
                  EXPRESSION(FSYS+[RPARENT]);
                  LOAD;
                  CASE LKEY OF
                  1: ORD;
                  2: CHR;
                  3: ODD
                  END (* CASE *);
                  INTEST(RPARENT,4)
                END
              ELSE
                IF LKEY=4 THEN EOLN ELSE EOF
        END (* STANDARD PROCEDURE AND FUNCTIONS *)
      ELSE CALLNONSTANDARD
    END (* CALL *);

    PROCEDURE EXPRESSION;
    VAR LATTR: ATTR; LOP: OPERATOR;

      PROCEDURE SIMPLEEXPRESSION(FSYS: SETOFSYS);
      VAR LATTR: ATTR; LOP: OPERATOR; SIGNED: BOOLEAN;

        PROCEDURE OPCOMP(VAR LATTR: ATTR; OPI,OPB,OPS: INTEGER);
        (* PROCEDURE TO COMPILE OPERATIONS.
           OPI,OPB,OPS ARE THE OPERATIONS TO GENERATE FOR OPERANDS OF
           TYPE INTEGER, BOOLEAN AND SET RESPECTIVELY.  AN AGRUMENT
           WILL BE ZERO FOR OPERAND/OPERATOR PAIRS WHICH ARE
           INVALID. *)
        VAR TEST: BOOLEAN;
        BEGIN TEST:=FALSE;
          IF (COMPTYPES(LATTR.TYPTR,INTPTR) AND
          COMPTYPES(GATTR.TYPTR,INTPTR)) THEN
            IF OPI<>0 THEN BEGIN TEST:=TRUE; BYTEGEN(OPI) END
            ELSE
          ELSE
            IF (COMPTYPES(LATTR.TYPTR,BOOLPTR) AND
            COMPTYPES(GATTR.TYPTR,BOOLPTR)) THEN
              IF OPB<>0 THEN BEGIN TEST:=TRUE; BYTEGEN(OPB) END
              ELSE
             ELSE
              IF (LATTR.TYPTR↑.FORM=POWER) AND
              COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
                IF OPS<>0 THEN BEGIN TEST:=TRUE; BYTEGEN(OPS) END;
          IF NOT TEST THEN BEGIN ERROR(134); GATTR.TYPTR:=NIL END
        END (* OPCOMP *);

        PROCEDURE TERM(FSYS: SETOFSYS);
        VAR LATTR: ATTR; LOP: OPERATOR;

          PROCEDURE FACTOR(FSYS: SETOFSYS);
          VAR LCP: CTP; LVP: CSP; VARPART: BOOLEAN;
```

```
17          CSTPART: INTSET; LSP: STP;
1800
1801        PROCEDURE SETEXPRESSION;
1802        BEGIN INSYMBOL; CSTPART:=[]; VARPART:=FALSE;
1803          NEW(LSP);
1804          LSP↑.STYPE :=NIL; LSP↑.SIZE :=SETSIZE;LSP↑.FORM:=POWER;
1805          IF SY=RBRACK THEN
1806            BEGIN
1807              GATTR.TYPTR:=LSP; GATTR.KIND:=CST;
1808              INSYMBOL
1809            END
1810          ELSE
1811            BEGIN
1812              REPEAT EXPRESSION(FSYS+[COMMA,RBRACK]);
1813                IF GATTR.TYPTR<>NIL THEN
1814                  IF GATTR.TYPTR↑.FORM>SUBRANGE THEN
1815                    BEGIN ERROR(136); GATTR.TYPTR:=NIL END
1816                  ELSE
1817                    IF COMPTYPES(LSP↑.STYPE,GATTR.TYPTR) THEN
1818                      BEGIN
1819                        IF GATTR.KIND=CST THEN
1820                          CSTPART:=CSTPART+[GATTR.CVAL.IVAL]
1821                        ELSE
1822                          BEGIN LOAD; BYTEGEN(174(*SGS*));
1823                            IF VARPART THEN BYTEGEN(175(*UNI*))
1824                            ELSE VARPART:=TRUE
1825                          END;
1826                        LSP↑.STYPE:=GATTR.TYPTR;
1827                        GATTR.TYPTR:=LSP
1828                      END
1829                    ELSE ERROR(137);
1830                TEST:=SY<>COMMA;
1831                IF NOT TEST THEN INSYMBOL
1832              UNTIL TEST;
1833              INTEST(RBRACK,12)
1834            END;
1835          IF VARPART THEN
1836            BEGIN
1837              IF CSTPART<>[] THEN
1838                BEGIN LDCS(CSTPART);
1839                  BYTEGEN(175(*UNI*));
1840                  GATTR.KIND:=EXPR
1841                END
1842            END
1843          ELSE
1844            BEGIN NEW(LVP); LVP↑.PVAL:=CSTPART;
1845              GATTR.CVAL.VALP:=LVP
1846            END
1847        END (* SETEXPRESSION *);
1848
1849      BEGIN (* FACTOR *)
1850        IF NOT(SY IN FACBEGSYS) THEN
1851          BEGIN ERROR(58); SKIP(FSYS+FACBEGSYS);
1852            GATTR.TYPTR:=NIL
1853          END;
1854        WHILE SY IN FACBEGSYS DO
1855          BEGIN
1856            CASE SY OF
1857              IDENT:
1858                BEGIN SEARCHID([KONST,VARS,FUNC],LCP);
1859                  INSYMBOL;
1860                  IF LCP↑.KLASS=FUNC THEN
```

```
                    BEGIN CALL(FSYS,LCP); GATTR.KIND:=EXPR END
                  ELSE
                    IF LCP↑.KLASS=KONST THEN
                      BEGIN GATTR.TYPTR:=LCP↑.IDTYPE;
                        GATTR.KIND:=CST;
                        GATTR.CVAL:=LCP↑.VALUES
                      END
                    ELSE
                      BEGIN SELECTOR(FSYS,LCP);
                      END
                END;
              INTCONST:
                BEGIN
                  GATTR.TYPTR:=INTPTR; GATTR.KIND:=CST;
                  GATTR.CVAL:=VAL;
                  INSYMBOL
                END;
              STRINGCONST:
                BEGIN
                  IF LGTH=1 THEN GATTR.TYPTR:=CHARPTR
                  ELSE
                    BEGIN NEW(LSP);
                      LSP↑.STYPE:=CHARPTR; LSP↑.FORM:=ARRAYS;
                      LSP↑.INXTYPE:=NIL;
                      LSP↑.SIZE:=STRINGSIZE(VAL.VALP);
                      GATTR.TYPTR:=LSP
                    END;
                  GATTR.KIND:=CST; GATTR.CVAL:=VAL;
                  INSYMBOL
                END;
              LPARENT:
                BEGIN INSYMBOL; EXPRESSION(FSYS+[RPARENT]);
                  INTEST(RPARENT,4)
                END;
              NOTSY:
                BEGIN INSYMBOL; FACTOR(FSYS); LOAD;
                  BYTEGEN(172(*NOT*));
                  IF GATTR.TYPTR<>NIL THEN
                    IF NOT COMPTYPES(GATTR.TYPTR,BOOLPTR) THEN
                      BEGIN ERROR(135); GATTR.TYPTR:=NIL END;
                END;
              LBRACK: SETEXPRESSION
            END;
            TEST2(FSYS,6,FACBEGSYS)
          END (* WHILE *)
      END (* FACTOR *);

    BEGIN (* TERM *)
      FACTOR(FSYS+[MULOP]);
      WHILE SY=MULOP DO
        BEGIN LOAD; LATTR:=GATTR; LOP:=OP;
          INSYMBOL; FACTOR(FSYS+[MULOP]); LOAD;
          IF (LATTR.TYPTR<>NIL) AND (GATTR.TYPTR<>NIL) THEN
            IF LOP=MUL THEN OPCOMP(LATTR,170(*MPI*),0,167(*INT*))
            ELSE
              IF LOP=IDIV THEN OPCOMP(LATTR,165(*DVI*),0,0)
              ELSE
                IF LOP=IMOD THEN OPCOMP(LATTR,169(*MOD*),0,0)
                ELSE
                  IF LOP=ANDOP THEN OPCOMP(LATTR,0,163(*AND*),0)
                  ELSE
          ELSE GATTR.TYPTR:=NIL
```

```
1923            END (* WHILE *)
1924        END (* TERM *);
1925
1926    BEGIN (* SIMPLEEXPRESSION *)
1927      SIGNED:=FALSE;
1928      IF (SY=ADDOP) AND (OP IN [PLUS,MINUS]) THEN
1929        BEGIN SIGNED:=OP=MINUS; INSYMBOL END;
1930      TERM(FSYS+[ADDOP]);
1931      IF SIGNED THEN
1932        BEGIN LOAD;
1933          IF COMPTYPES(GATTR.TYPTR,INTPTR) THEN BYTEGEN(171(*NGI*))
1934          ELSE
1935            BEGIN ERROR(134); GATTR.TYPTR:=NIL END
1936        END;
1937      WHILE SY=ADDOP DO
1938        BEGIN LOAD; LATTR:=GATTR; LOP:=OP;
1939          INSYMBOL; TERM(FSYS+[ADDOP]); LOAD;
1940          IF (LATTR.TYPTR<>NIL) AND (GATTR.TYPTR<>NIL) THEN
1941            IF LOP=PLUS THEN OPCOMP(LATTR,162(*ADI*),0,175(*UNI*))
1942            ELSE
1943              IF LOP=MINUS THEN OPCOMP(LATTR,173(*SBI*),0,164(*DIF*))
1944              ELSE
1945                IF LOP=OROP THEN OPCOMP(LATTR,0,168(*IOR*),0)
1946                ELSE
1947          ELSE GATTR.TYPTR:=NIL;
1948          GATTR.KIND:=EXPR
1949        END (* WHILE *)
1950    END (* SIMPLEEXPRESSION *);
1951
1952  BEGIN (* EXPRESSION *)
1953    SIMPLEEXPRESSION(FSYS+[RELOP]);
1954    IF SY=RELOP THEN
1955      BEGIN
1956        IF GATTR.TYPTR<>NIL THEN
1957          IF GATTR.TYPTR↑.FORM<=POWER THEN LOAD
1958          ELSE LOADADDRESS;
1959        LATTR:=GATTR; LOP:=OP;
1960        INSYMBOL; SIMPLEEXPRESSION(FSYS);
1961        IF GATTR.TYPTR<>NIL THEN
1962          IF GATTR.TYPTR↑.FORM<=POWER THEN LOAD
1963          ELSE LOADADDRESS;
1964        IF (LATTR.TYPTR<>NIL) AND (GATTR.TYPTR<>NIL) THEN
1965          IF LOP=INOP THEN
1966            IF GATTR.TYPTR↑.FORM=POWER THEN
1967              IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR↑.STYPE) THEN
1968                BYTEGEN(166(*INN*))
1969              ELSE BEGIN ERROR(129); GATTR.TYPTR:=NIL END
1970            ELSE BEGIN ERROR(130); GATTR.TYPTR:=NIL END
1971          ELSE
1972            BEGIN
1973              IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
1974                BEGIN
1975                  IF LATTR.TYPTR↑.FORM=POINTER THEN
1976                    BEGIN
1977                      IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
1978                    END
1979                  ELSE
1980                    IF LATTR.TYPTR↑.FORM=POWER THEN
1981                      BEGIN
1982                        IF LOP IN [LTOP,LEOP,GEOP,GTOP] THEN ERROR(132)
1983                      END
1984                    ELSE
```

```
                        IF LATTR.TYPTR↑.FORM=ARRAYS THEN
                          BEGIN
                            IF NOT STRING(LATTR.TYPTR) AND
                              (LOP IN [LTOP,LEOP,GTOP,GEOP]) THEN
                                ERROR(131)
                          END
                        ELSE
                          IF LATTR.TYPTR↑.FORM=RECORDS THEN
                            IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN
                              ERROR(131);
                    IF LOP=LTOP THEN CONDGEN(147(*LES*),LATTR)
                    ELSE
                      IF LOP=LEOP THEN CONDGEN(144(*LEQ*),LATTR)
                      ELSE
                        IF LOP=EQOP THEN CONDGEN(150(*EQU*),LATTR)
                        ELSE
                          BEGIN
                            IF LOP=GTOP THEN CONDGEN(144(*LEQ*),LATTR)
                            ELSE
                              IF LOP=GEOP THEN CONDGEN(147(*LES*),LATTR)
                              ELSE
                                IF LOP=NEOP THEN CONDGEN(150(*EQU*),LATTR);
                            BYTEGEN(172(*NOT*))
                          END
                  END
                ELSE ERROR(129)
            END;
        GATTR.TYPTR:=BOOLPTR; GATTR.KIND:=EXPR
      END (* SY=RELOP *)
  END (*EXPRESSION *);

  PROCEDURE ASSIGNMENT(FCP: CTP);
  VAR LATTR: ATTR;
  BEGIN SELECTOR(FSYS+[BECOMES],FCP);
    IF SY=BECOMES THEN
      BEGIN
        IF GATTR.TYPTR<>NIL THEN
          IF (GATTR.ACCESS<>DRCT) OR (GATTR.TYPTR↑.FORM>POWER) THEN
            LOADADDRESS;
        LATTR:=GATTR;
        INSYMBOL; EXPRESSION(FSYS);
        IF GATTR.TYPTR<>NIL THEN
          IF GATTR.TYPTR↑.FORM<=POWER THEN LOAD
          ELSE LOADADDRESS;
        IF (LATTR.TYPTR<>NIL) AND (GATTR.TYPTR<>NIL) THEN
          BEGIN
            IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
          IF LATTR.TYPTR↑.FORM IN [ARRAYS,RECORDS] THEN
                BEGIN BYTEGEN(183(*MOV*)); ADDRGEN(LATTR.TYPTR↑.SIZE) END
              ELSE STORE(LATTR)
            ELSE ERROR(129)
          END
      END (* SY=BECOMES *)
    ELSE ERROR(51)
  END (* ASSIGNMENT *);

  PROCEDURE COMPOUNDSTATEMENT;
  BEGIN
    REPEAT
      REPEAT STATEMENT(FSYS+[SEMICOLON,ENDSY])
      UNTIL NOT(SY IN STATBEGSYS);
      TEST:=SY<>SEMICOLON;
```

```
2047      IF NOT TEST THEN INSYMBOL
2048    UNTIL TEST;
2049    INTEST(ENDSY,13)
2050  END (* COMPOUNDSTATEMENT *);
2051
2052  PROCEDURE IFSTATEMENT;
2053  VAR ICIX1,ICIX2: INTEGER;
2054  BEGIN EXPRESSION(FSYS+[THENSY]);
2055    GENUJP(0); ICIX1:=IC+ICN-2;
2056    INTEST(THENSY,52);
2057    STATEMENT(FSYS+[ELSESY]);
2058    IF SY=ELSESY THEN
2059      BEGIN ICIX2:=IC+ICN; GENUJPENT(178(*UJP*),0);
2060        PLANTADDR(ICIX1,IC+ICN);
2061        INSYMBOL; STATEMENT(FSYS);
2062        PLANTADDR(ICIX2+1,IC+ICN);
2063      END
2064    ELSE PLANTADDR(ICIX1,IC+ICN)
2065  END (* IFSTATEMENT *);
2066
2067  PROCEDURE CASESTATEMENT;
2068  TYPE CIP=↑CASEINFO;
2069    CASEINFO=PACKED RECORD NEXT: CIP;
2070                CSSTART,CSLAB: INTEGER
2071              END;
2072  VAR LSP,LSP1: STP; FSTPTR,LPT1,LPT2,LPT3: CIP; LVAL: VALU;
2073      LCIX,LCIX1,LMIN,LMAX: INTEGER;
2074  BEGIN EXPRESSION(FSYS+[OFSY,COMMA,COLON]); LOAD;
2075    LCIX:=IC+ICN; GENUJPENT(178(*UJP*),0); GENUJPENT(178(*UJP*),0);
2076    LSP:=GATTR.TYPTR;
2077    IF LSP<>NIL THEN
2078      IF LSP↑.FORM>SUBRANGE THEN
2079        BEGIN ERROR(144); LSP:=NIL END;
2080    INTEST(OFSY,8);
2081    FSTPTR:=NIL;
2082    REPEAT LPT3:=NIL; LCIX1:=IC+ICN;
2083      REPEAT CONSTANT(FSYS+[COMMA,COLON],LSP1,LVAL);
2084        IF LSP1<>NIL THEN
2085          IF COMPTYPES(LSP,LSP1) THEN
2086            BEGIN LPT1:=FSTPTR; LPT2:=NIL; TEST:=TRUE;
2087              WHILE (LPT1<>NIL) AND TEST DO
2088                IF LPT1↑.CSLAB<=LVAL.IVAL THEN
2089                  BEGIN IF LPT1↑.CSLAB=LVAL.IVAL THEN ERROR(156);
2090                    TEST:=FALSE
2091                  END
2092                ELSE BEGIN LPT2:=LPT1; LPT1:=LPT1↑.NEXT END;
2093              NEW(LPT3); LPT3↑.NEXT:=LPT1; LPT3↑.CSLAB:=LVAL.IVAL;
2094              LPT3↑.CSSTART:=LCIX1;
2095              IF LPT2=NIL THEN FSTPTR:=LPT3
2096              ELSE LPT2↑.NEXT:=LPT3
2097            END
2098          ELSE ERROR(147);
2099        TEST:=SY<>COMMA;
2100        IF NOT TEST THEN INSYMBOL;
2101      UNTIL TEST;
2102      INTEST(COLON,5);
2103      REPEAT STATEMENT(FSYS+[SEMICOLON,ELSESY])
2104      UNTIL NOT(SY IN STATBEGSYS);
2105      IF LPT3<>NIL THEN GENUJPENT(178(*UJP*),LCIX+3);
2106      TEST:=SY<>SEMICOLON;
2107      IF NOT TEST THEN INSYMBOL;
2108    UNTIL TEST OR (SY=ENDSY) OR (SY=ELSESY);
```

```
2109    PLANTADDR(LCIX+1,IC+ICN);
2110    IF FSTPTR<>NIL THEN
2111      BEGIN LMAX:=FSTPTR↑.CSLAB; LPT1:=FSTPTR; FSTPTR:=NIL;
2112        (* REVERSE POINTERS *)
2113        REPEAT LPT2:=LPT1↑.NEXT; LPT1↑.NEXT:=FSTPTR;
2114          FSTPTR:=LPT1; LPT1:=LPT2
2115        UNTIL LPT1=NIL;
2116        LMIN:=FSTPTR↑.CSLAB;
2117        BYTEGEN(182(*CAS*)); ADDRGEN(LMIN); ADDRGEN(LMAX);
2118        LCIX1:=IC+ICN; ADDRGEN(0);
2119        REPEAT
2120          WHILE FSTPTR↑.CSLAB>LMIN DO
2121            BEGIN ADDRGEN(0); LMIN:=LMIN+1 END;
2122          ADDRGEN(FSTPTR↑.CSSTART);
2123          FSTPTR:=FSTPTR↑.NEXT; LMIN:=LMIN+1
2124        UNTIL FSTPTR=NIL
2125      END
2126    ELSE ERROR(157);
2127    IF SY=ELSESY THEN
2128      BEGIN INSYMBOL; PLANTADDR(LCIX1,IC+ICN);
2129        REPEAT STATEMENT(FSYS+[SEMICOLON])
2130        UNTIL NOT(SY IN STATBEGSYS);
2131        GENUJPENT(178(*UJP*),LCIX+3)
2132      END;
2133    PLANTADDR(LCIX+4,IC+ICN);
2134    INTEST(ENDSY,13)
2135  END (* CASESTATEMENT *);
2136
2137  PROCEDURE REPEATSTATEMENT;
2138  VAR LADDR: INTEGER;
2139  BEGIN LADDR:=IC+ICN;
2140    REPEAT
2141      REPEAT STATEMENT(FSYS+[SEMICOLON,UNTILSY])
2142      UNTIL NOT(SY IN STATBEGSYS);
2143      TEST:=SY<>SEMICOLON;
2144      IF NOT TEST THEN INSYMBOL
2145    UNTIL TEST;
2146    IF SY=UNTILSY THEN
2147      BEGIN INSYMBOL; EXPRESSION(FSYS); GENFJP(LADDR)
2148      END
2149    ELSE ERROR(53)
2150  END (* REPEATSTATEMENT *);
2151
2152  PROCEDURE WHILESTATEMENT;
2153  VAR LADDR,LCIX: INTEGER;
2154  BEGIN LADDR:=IC+ICN;
2155    EXPRESSION(FSYS+[DOSY]); LCIX:=IC+ICN; GENFJP(0);
2156    INTEST(DOSY,54);
2157    STATEMENT(FSYS); GENUJPENT(178(*UJP*),LADDR); PLANTADDR(LCIX+1,IC+ICN)
2158  END (* WHILESTATEMENT *);
2159
2160  PROCEDURE FORSTATEMENT;
2161  VAR LATTR: ATTR; LSP: STP; LSY: SYMBOL; LCIX,LADDR: INTEGER;
2162  BEGIN
2163    IF SY=IDENT THEN
2164      BEGIN SEARCHID([VARS],LCP);
2165        LATTR.TYPTR:=LCP↑.IDTYPE; LATTR.KIND:=VARBL;
2166        IF LCP↑.VKIND=ACTUAL THEN
2167          LOADGEN(LEVEL-LCP↑.VLEV,LCP↑.VADDR)
2168        ELSE BEGIN ERROR(155); LATTR.TYPTR:=NIL END;
2169        IF LATTR.TYPTR<>NIL THEN
2170          IF (LATTR.TYPTR↑.FORM>SUBRANGE) (*OR LATTR.TYPTR=REALPTR*) THEN
```

```
2171            BEGIN ERROR(143); LATTR.TYPTR:=NIL END;
2172          INSYMBOL
2173        END
2174      ELSE
2175        BEGIN ERROR(2); SKIP(FSYS+[BECOMES,TOSY,DOWNTOSY,DOSY]) END;
2176      IF SY=BECOMES THEN
2177        BEGIN INSYMBOL; EXPRESSION(FSYS+[TOSY,DOWNTOSY,DOSY]);
2178          IF GATTR.TYPTR<>NIL THEN
2179            IF GATTR.TYPTR↑.FORM>SUBRANGE THEN ERROR(144)
2180            ELSE
2181              IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN LOAD
2182              ELSE ERROR(145)
2183        END
2184      ELSE
2185        BEGIN ERROR(51); SKIP(FSYS+[TOSY,DOWNTOSY,DOSY]) END;
2186      IF SY IN [TOSY,DOWNTOSY] THEN
2187        BEGIN LSY:=SY; INSYMBOL; EXPRESSION(FSYS+[DOSY]);
2188          IF GATTR.TYPTR<>NIL THEN
2189            IF GATTR.TYPTR↑.FORM>SUBRANGE THEN ERROR(144)
2190            ELSE
2191              IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
2192                BEGIN LOAD; LCIX:=LATTR.TYPTR↑.SIZE-1;
2193                  IF LSY=DOWNTOSY THEN LCIX:=LCIX+4;
2194                  BYTEGEN(LCIX); LADDR:=IC+ICN;
2195                  GENUJPENT(178(*UJP*),0)
2196                END
2197              ELSE ERROR(145)
2198        END
2199      ELSE BEGIN ERROR(55); SKIP(FSYS+[DOSY]) END;
2200      INTEST(DOSY,54);
2201      STATEMENT(FSYS);
2202      PLANTADDR(LADDR+1,IC+ICN);
2203      GENUJPENT(145(*FOR*),LADDR+3)
2204    END (* FORSTATEMENT *);
2205
2206    BEGIN (* STATEMENT *)
2207      IF NOT(SY IN FSYS+[IDENT]) THEN
2208        BEGIN ERROR(6); SKIP(FSYS) END;
2209      IF SY IN STATBEGSYS+[IDENT] THEN
2210        BEGIN
2211          PFCTTEST:=FALSE;
2212          CASE SY OF
2213            IDENT: BEGIN
2214              SEARCHID([VARS,FUNC,PROC],LCP); INSYMBOL;
2215              IF LCP↑.KLASS=PROC THEN CALL(FSYS,LCP)
2216              ELSE ASSIGNMENT(LCP)
2217            END;
2218            BEGINSY: BEGIN INSYMBOL; COMPOUNDSTATEMENT END;
2219            IFSY: BEGIN INSYMBOL; IFSTATEMENT END;
2220            WHILESY: BEGIN INSYMBOL; WHILESTATEMENT END;
2221            REPEATSY: BEGIN INSYMBOL; REPEATSTATEMENT END;
2222            FORSY: BEGIN INSYMBOL; FORSTATEMENT END;
2223            CASESY: BEGIN INSYMBOL; CASESTATEMENT END
2224          END;
2225          IF NOT(SY IN [SEMICOLON,ENDSY,ELSESY,UNTILSY]) THEN
2226            BEGIN ERROR(6); SKIP(FSYS) END
2227        END
2228    END (* STATEMENT *);
2229
2230    BEGIN (* BODY *)
2231      ENTNAME:=FPROCP↑.PFNAME;
2232      WRITEOUT; SUMCHECK:=0; WRITE(= P4=); HEXOUT(ENTNAME);
```

```
2233    FOR I:=1 TO 8 DO
2234      BEGIN WRITE(FPROCP↑.NAME[I]);
2235        SUMCHECK:=SUMCHECK+ORD(FPROCP↑.NAME[I])
2236      END;
2237    HEXOUT((16383-SUMCHECK) MOD 256); WRITELN;
2238    SEGSIZE:=IC+ICN;
2239    GENUJPENT(181(*ENT*),0);
2240    LLC1:=LCAFTERMARKSTACK; LCP:=FPROCP↑.NEXT;
2241    WHILE LCP<>NIL DO
2242      BEGIN
2243        IF LCP↑.KLASS=VARS THEN
2244          IF LCP↑.IDTYPE<>NIL THEN
2245            IF (LCP↑.VKIND=ACTUAL)AND(LCP↑.IDTYPE↑.FORM>POWER) THEN
2246              BEGIN LLC1:=LLC1+PTRSIZE; LDAGEN(0,LCP↑.VADDR);
2247                BYTEGEN(96); BYTEGEN(LLC1); BYTEGEN(183(*MOV*));
2248                ADDRGEN(LCP↑.IDTYPE↑.SIZE)
2249              END
2250            ELSE LLC1:=LLC1+LCP↑.IDTYPE↑.SIZE;
2251        LCP:=LCP↑.NEXT;
2252      END;
2253    LCMAX:=LC;
2254    REPEAT
2255      REPEAT STATEMENT(FSYS+[SEMICOLON,ENDSY])
2256      UNTIL NOT(SY IN STATBEGSYS);
2257      TEST:=SY<>SEMICOLON;
2258      IF NOT TEST THEN INSYMBOL
2259    UNTIL TEST;
2260    INTEST(ENDSY,13);
2261    BYTEGEN(161(*RETP*)); PLANTADDR(SEGSIZE+1,LCMAX-6);
2262  END (* BODY *);
2263
2264  BEGIN (* BLOCK *)
2265    DP:=TRUE;
2266    REPEAT
2267      IF SY=CONSTSY THEN
2268        BEGIN INSYMBOL; CONSTDECLARATION END;
2269      IF SY=TYPESY THEN
2270        BEGIN INSYMBOL; TYPEDECLARATION END;
2271      IF SY=VARSY THEN
2272        BEGIN INSYMBOL; VARDECLARATION END;
2273      WHILE SY IN [PROCSY,FUNCSY] DO
2274        BEGIN LSY:=SY; INSYMBOL; PROCDECLARATION(LSY) END;
2275      IF SY<>BEGINSY THEN
2276        BEGIN ERROR(18); SKIP(FSYS) END
2277    UNTIL SY IN STATBEGSYS;
2278    DP:=FALSE;
2279    INTEST(BEGINSY,17);
2280    REPEAT BODY(FSYS);
2281      TEST2([FSY],6,FSYS)
2282    UNTIL (SY=FSY) OR (SY IN BLOCKBEGSYS);
2283  END (* BLOCK *);
2284
2285  PROCEDURE STDNAMES;
2286  BEGIN
2287    NA[ 0]:='FALSE    '; NA[ 1]:='TRUE     '; NA[ 2]:='READ     ';
2288    NA[3]:='WRITE    ';
2289    NA[4]:='NEW      '; NA[5]:='RELEASE  '; NA[6]:='READLN   ';
2290    NA[7]:='WRITELN '; NA[8]:='ORD      '; NA[9]:='CHR      ';
2291    NA[10]:='ODD      '; NA[11]:='EOLN     '; NA[12]:='EOF      ';
2292  END (* STDNAMES *);
2293
2294  PROCEDURE ENTERSTDTYPES;
```

```
2295 BEGIN
2296   NEW(INTPTR);  (* INTEGER *)
2297   INTPTR↑.SIZE:=INTSIZE; INTPTR↑.FORM:=SCALAR; INTPTR↑.STYPE:=NIL;
2298   NEW(CHARPTR); (* CHAR *)
2299   CHARPTR↑.SIZE:=CHARSIZE; CHARPTR↑.FORM:=SCALAR; CHARPTR↑.STYPE:=NIL;
2300   NEW(BOOLPTR); (* BOOLEAN *)
2301   BOOLPTR↑.SIZE:=BOOLSIZE; BOOLPTR↑.FORM:=SCALAR; BOOLPTR↑.STYPE:=NIL;
2302   NEW(NILPTR);           (* NIL *)
2303   NILPTR↑.STYPE:=NIL; NILPTR↑.SIZE:=PTRSIZE; NILPTR↑.FORM:=POINTER;
2304 END (* ENTERSTDTYPES *);
2305
2306 PROCEDURE ENTSTDNAMES;
2307 VAR CP,CP1: CTP; I: INTEGER;
2308 BEGIN
2309   NEW(CP);      (* INTEGER *)
2310   CP↑.NAME:='INTEGER '; CP↑.IDTYPE:=INTPTR; CP↑.KLASS:=TYPES;
2311   ENTERID(CP);
2312   NEW(CP);      (* CHAR *)
2313   CP↑.NAME:='CHAR    '; CP↑.IDTYPE:=CHARPTR; CP↑.KLASS:=TYPES;
2314   ENTERID(CP);
2315   NEW(CP);      (* BOOLEAN *)
2316   CP↑.NAME:='BOOLEAN '; CP↑.IDTYPE:=BOOLPTR; CP↑.KLASS:=TYPES;
2317   ENTERID(CP);
2318   CP1:=NIL;
2319   FOR I:=0 TO 1 DO
2320     BEGIN NEW(CP);    (* FALSE,TRUE *)
2321       CP↑.NAME:=NA[I+1]; CP↑.IDTYPE:=BOOLPTR; CP↑.NEXT:=CP1;
2322       CP↑.VALUES.IVAL:=I; CP↑.KLASS:=KONST;
2323       ENTERID(CP); CP1:=CP
2324     END;
2325   NEW(CP);      (* NIL *)
2326   CP↑.NAME:='NIL     '; CP↑.IDTYPE:=NILPTR;
2327   CP↑.NEXT:=NIL; CP↑.VALUES.IVAL:=0; CP↑.KLASS:=KONST;
2328   ENTERID(CP);
2329   FOR I:=2 TO 7 DO
2330     BEGIN NEW(CP);
2331       CP↑.NAME:=NA[I]; CP↑.IDTYPE:=NIL;
2332       CP↑.NEXT:=NIL; CP↑.KEY:=I-1;
2333       CP↑.KLASS:=PROC; CP↑.PFDECKIND:=STANDARD;
2334       ENTERID(CP)
2335     END;
2336   NEW(CP); CP↑.NAME:='EXIT    '; CP↑.IDTYPE:=NIL; CP↑.NEXT:=NIL;
2337   CP↑.KEY:=8; CP↑.KLASS:=PROC; CP↑.PFDECKIND:=STANDARD;
2338   ENTERID(CP);
2339   FOR I:=8 TO 12 DO
2340     BEGIN NEW(CP);
2341       CP↑.NAME:=NA[I]; CP↑.IDTYPE:=NIL; CP↑.NEXT:=NIL;
2342       CP↑.KEY:=I-7; CP↑.KLASS:=FUNC;
2343       CP↑.PFDECKIND:=STANDARD;
2344       ENTERID(CP)
2345     END
2346 END (* ENTSTDNAMES *);
2347
2348 PROCEDURE ENTERUNDECL;
2349 BEGIN
2350   NEW(UTYPPTR);
2351   UTYPPTR↑.NAME:='        '; UTYPPTR↑.IDTYPE:=NIL;
2352   UTYPPTR↑.KLASS:=TYPES;
2353   NEW(UCSTPTR);
2354   UCSTPTR↑.NAME:='        '; UCSTPTR↑.IDTYPE:=NIL; UCSTPTR↑.NEXT:=NIL;
2355   UCSTPTR↑.VALUES.IVAL:=0; UCSTPTR↑.KLASS:=KONST;
2356   NEW(UVARPTR);
```

```
2357    UVARPTR↑.NAME:=≡          ≡; UVARPTR↑.IDTYPE:=NIL;
2358    UVARPTR↑.VKIND:=ACTUAL;
2359    UVARPTR↑.NEXT:=NIL; UVARPTR↑.VLEV:=0; UVARPTR↑.VADDR:=0;
2360    UVARPTR↑.KLASS:=VARS;
2361    NEW(UFLDPTR);
2362    UFLDPTR↑.NAME:=≡          ≡; UFLDPTR↑.IDTYPE:=NIL; UFLDPTR↑.NEXT:=NIL;
2363    UFLDPTR↑.VADDR:=0; UFLDPTR↑.KLASS:=FIELD;
2364    NEW(UPRCPTR);
2365    UPRCPTR↑.NAME:=≡          ≡; UPRCPTR↑.IDTYPE:=NIL;
2366    UPRCPTR↑.FORWDECL:=FALSE; UPRCPTR↑.EXTERNL:=FALSE;
2367    UPRCPTR↑.NEXT:=NIL; UPRCPTR↑.PFLEV:=0; UPRCPTR↑.PFNAME:=0;
2368    UPRCPTR↑.KLASS:=PROC; UPRCPTR↑.PFDECKIND:=DECLARED;
2369    NEW(UFCTPTR);
2370    UFCTPTR↑.NAME:=≡          ≡; UFCTPTR↑.IDTYPE:=NIL; UFCTPTR↑.NEXT:=NIL;
2371    UFCTPTR↑.FORWDECL:=FALSE; UFCTPTR↑.PFLEV:=0; UFCTPTR↑.PFNAME:=0;
2372    UFCTPTR↑.EXTERNL:=FALSE;
2373    UFCTPTR↑.KLASS:=FUNC; UFCTPTR↑.PFDECKIND:=DECLARED
2374  END (* ENTERUNDECL *);
2375
2376  BEGIN (* INITIALIZE *)
2377     SETHSP; (* OUTPUT TO HSP *)
2378   SETHSR;(*READ HSR *)
2379     ERRFLAG := FALSE ; (* INIT ERR FLAG *)
2380     LINECOUNT:=0;      (*ADD LINE NUMBERS*)
2381    FWPTR:=NIL;
2382    DP:=TRUE; PRTERR:=TRUE; ERRINX:=0;
2383    LC:=LCAFTERMARKSTACK;
2384    IC:=0; ICN:=0; CH:=≡ ≡; ICH:=32;
2385    BEGINLINE;
2386    MXINT10:=MAXINT DIV 10;
2387    NPROC:=1;
2388
2389    (* INITIALIZE SETS *)
2390    CONSTBEGSYS:=[ADDOP,INTCONST,STRINGCONST,IDENT];
2391    SIMPTYPEBEGSYS:=[LPARENT]+CONSTBEGSYS;
2392    TYPEBEGSYS:=[ARROW,PACKEDSY,ARRAYSY,RECORDSY,SETSY]+
2393      SIMPTYPEBEGSYS;
2394    TYPEDELS:=[ARRAYSY,RECORDSY,SETSY];
2395    BLOCKBEGSYS:=[CONSTSY,TYPESY,VARSY,PROCSY,FUNCSY,
2396      BEGINSY];
2397    SELECTSYS:=[ARROW,PERIOD,LBRACK];
2398    FACBEGSYS:=[INTCONST,STRINGCONST,IDENT,LPARENT,
2399      REALCONST,LBRACK,NOTSY];
2400    STATBEGSYS:=[BEGINSY,IFSY,WHILESY,REPEATSY,CASESY,FORSY];
2401
2402    (* INITIALIZE TABLES *)
2403    RW[ 0]:=≡IF       ≡; RW[ 1]:=≡DO       ≡; RW[ 2]:=≡OF       ≡;
2404    RW[3]:=≡IN       ≡; RW[4]:=≡OR       ≡; RW[5]:=≡TO       ≡;
2405    RW[6]:=≡END      ≡; RW[7]:=≡FOR      ≡; RW[8]:=≡VAR      ≡;
2406    RW[9]:=≡DIV      ≡; RW[10]:=≡MOD      ≡; RW[11]:=≡SET      ≡;
2407    RW[12]:=≡AND      ≡; RW[13]:=≡NOT      ≡; RW[14]:=≡THEN     ≡;
2408    RW[15]:=≡ELSE     ≡; RW[16]:=≡TYPE     ≡; RW[17]:=≡CASE     ≡;
2409    RW[18]:=≡BEGIN    ≡; RW[19]:=≡UNTIL    ≡; RW[20]:=≡WHILE    ≡;
2410    RW[21]:=≡ARRAY    ≡; RW[22]:=≡CONST    ≡; RW[23]:=≡REPEAT   ≡;
2411    RW[24]:=≡RECORD   ≡; RW[25]:=≡PACKED   ≡; RW[26]:=≡EXTERN   ≡;
2412    RW[27]:=≡DOWNTO   ≡; RW[28]:=≡FORWARD  ≡; RW[29]:=≡PROGRAM  ≡;
2413    RW[30]:=≡FUNCTION≡; RW[31]:=≡PROCEDUR≡;
2414    FRW[0]:=0; FRW[1]:=0; FRW[2]:=6; FRW[3]:=14; FRW[4]:=18;
2415    FRW[5]:=23; FRW[6]:=26; FRW[7]:=30; FRW[8]:=32;
2416
2417    (* INITIALIZE SYMBOLS *)
2418    RSY[0]:=IFSY; RSY[1]:=DOSY; RSY[2]:=OFSY;
```

```
2419    RSY[3]:=RELOP; RSY[4]:=ADDOP; RSY[5]:=TOSY;
2420    RSY[6]:=ENDSY; RSY[7]:=FORSY; RSY[8]:=VARSY;
2421    RSY[9]:=MULOP; RSY[10]:=MULOP; RSY[11]:=SETSY;
2422    RSY[12]:=MULOP; RSY[13]:=NOTSY; RSY[14]:=THENSY;
2423    RSY[15]:=ELSESY; RSY[16]:=TYPESY; RSY[17]:=CASESY;
2424    RSY[18]:=BEGINSY; RSY[19]:=UNTILSY; RSY[20]:=WHILESY;
2425    RSY[21]:=ARRAYSY; RSY[22]:=CONSTSY; RSY[23]:=REPEATSY;
2426    RSY[24]:=RECORDSY; RSY[25]:=PACKEDSY; RSY[26]:=EXTERNSY;
2427    RSY[27]:=DOWNTOSY; RSY[28]:=FORWARDSY; RSY[29]:=PROGSY;
2428    RSY[30]:=FUNCSY; RSY[31]:=PROCSY;
2429    FOR I:=0 TO 127 DO
2430      BEGIN SSY[I]:=OTHERSY; SOP[I]:=NOOP END;
2431
2432    SSY[APLUS]:=ADDOP; SSY[AMINUS]:=ADDOP; SSY[ASTAR]:=MULOP;
2433    SSY[ASLASH]:=MULOP; SSY[ALPAREN]:=LPARENT; SSY[ARPAREN]:=RPARENT;
2434    SSY[ADOLLAR]:=OTHERSY; SSY[AEQUAL]:=RELOP; SSY[ASPACE]:=OTHERSY;
2435    SSY[ACOMMA]:=COMMA; SSY[APERIOD]:=PERIOD; SSY[ADQUOTE]:=OTHERSY;
2436    SSY[ALBRACK]:=LBRACK; SSY[ARBRACK]:=RBRACK; SSY[ACOLON]:=COLON;
2437    SSY[AARROW]:=ARROW; SSY[ALESS]:=RELOP; SSY[AGREATER]:=RELOP;
2438    SSY[ASEMI]:=SEMICOLON;
2439
2440    (* INITIALIZE OPERATORS *)
2441    FOR I:=0 TO 31 DO ROP[I]:=NOOP;
2442    ROP[3]:=INOP; ROP[9]:=IDIV; ROP[10]:=IMOD;
2443    ROP[4]:=OROP; ROP[12]:=ANDOP;
2444    SOP[APLUS]:=PLUS; SOP[AMINUS]:=MINUS; SOP[ASTAR]:=MUL;
2445    SOP[AEQUAL]:=EQOP; SOP[ASLASH]:=RDIV;
2446    SOP[ALESS]:=LTOP; SOP[AGREATER]:=GTOP;
2447
2448    (* ENTER STANDARD NAMES AND STANDARD TYPES *)
2449    (*****************************************)
2450    LEVEL:=0; TOP:=0;
2451    DISPLAY[0]:=NIL;
2452    ENTERSTDTYPES; STDNAMES; ENTSTDNAMES; ENTERUNDECL;
2453    TOP:=1; LEVEL:=1; SAVETOP:=1;
2454    DISPLAY[1]:=NIL;
2455
2456
2457    (* COMPILE *)
2458    (***********)
2459    INSYMBOL;
2460    NEW(PROGPTR); PROGPTR↑.IDTYPE:=NIL; PROGPTR↑.NEXT:=NIL;
2461    PROGPTR↑.KLASS:=PROC; PROGPTR↑.PFDECKIND:=DECLARED;
2462    PROGPTR↑.PFNAME:=0; PROGPTR↑.NAME:='          ';
2463    IF SY=PROGSY THEN
2464      BEGIN INSYMBOL;
2465        IF SY<>IDENT THEN ERROR(2) ELSE PROGPTR↑.NAME:=ID;
2466        WHILE SY<>SEMICOLON DO INSYMBOL;
2467        INSYMBOL
2468      END;
2469    BLOCK(BLOCKBEGSYS+STATBEGSYS,PERIOD,PROGPTR);
2470
2471    ENDLINE;
2472
2473    WRITEOUT; WRITELN('P9');
2474    IF ERRFLAG THEN BEGIN
2475      TTYOUT;
2476      WRITELN('COMPILE ERROR(S)') END;
2477  END.
```