```
0001:  (*$L+,C+,T-*)
0002:
0003:
0004:  (*******************************************************
0005:   *                                                     *
0006:   *                                                     *
0007:   *               PORTABLE COMPILER PROJECT             *
0008:   *               *************************             *
0009:   *                                                     *
0010:   *                      NEW EDITION                    *
0011:   *                   VERSION 16.02.77                  *
0012:   *                                                     *
0013:   *                                                     *
0014:   *                                                     *
0015:   *                                                     *
0016:   *     AUTHOR:   H. H. NAEGELI                         *
0017:   *               INSTITUT FUER INFORMATIK              *
0018:   *               EIDG. TECHNISCHE HOCHSCHULE           *
0019:   *       CH-8006 ZUERICH                               *
0020:   *                                                     *
0021:   *******************************************************)
0022:
0023:
0024:
0025:
0026:  (**)  (*THIS MARK AT THE BEGINNING OF A LINE INDICATES THAT SOME MACHINE
0027:          DEPENDENT CODE HAS TO BE INSERTED. A COMMENT DESCRIBES THE INSERTION*)
0028:
0029:
0030:
0031:
0032:
0033:  PROGRAM PASCAL(INPUT,OUTPUT
0034:  (**)                      (*.FILES. OTHER EXTERNAL FILES*) );
0035:
0036:
0037:  CONST
0038:            (*IMPLEMENTATION DEPENDENT CONSTANTS*)
0039:            (***********************************)
0040:
0041:  (**) DISPLIMIT =   (*MAX NUMBER OF NESTED SCOPES OF IDENTIFIERS*)     20;
0042:  (**) MAXLEVEL  =   (*MAX NUMBER OF NESTED PROC/FUNCT*)               10;
0043:  (**) MAXCHCNT  =   (* 1 + MAX NUMBER OF CHAR ON AN INPUT LINE*)      81;
0044:  (**) DIGMAX    =   (*MAX NUMBER OF DIGITS IN AN INTEGER NUMBER
0045:                       AND IN A REAL NUMBER BEFORE THE DECIMAL POINT *) 20;
0046:  (**) NSP       =   (*NUMBER OF CHAR WRITTEN OUT BY #WRCI# AND #WRN#
0047:                                                    TOGETHER*)   15;
0048:  (**) NGLAB     =   (*MAX NUMBER OF EXTERNAL LABELS*)                  58;
0049:
0050:            (*LANGUAGE CONSTANTS*)
0051:            (*******************)
0052:
0053:       RESWORDS  = 34;       (*NUMBER OF RESERVED WORDS*)
0054:       NRSTDPROC = 14; NRSTDFUNC = 11; NREXFUNC   = 6;
0055:                             (*NO OF STANDARD PROC,FUNC, OF STANDARD ARITH
0056:                               FUNCTIONS*)
0057:       ALFALENG  = 8;        (*NO OF SIGNIFICANT CHAR IN AN IDENTIFIER*)
0058:       NRSTDNAMES= 35;       (* = NRSTDPROC + NRSTDFUNC + NREXFUNC *)
0059:
0060:            (*MACHINE DEPENDENT CONSTANTS*)
0061:            (*****************************)
0062:
0063:  (**) MAXADDR   =   (*HIGHEST ADDRESS*)                          32767;
0064:  (**) ORDCHARMAX=   (*ORDINAL NUMBER OF THE LAST CHARACTER
0065:                                          (ON THE GOAL MACHINE) *) 63;
0066:  (**) SETMIN    =   (*SMALLEST ELEMENT OF A SET *)                  0;
```

```
0067:  (**) SETMAX    =   (*LARGEST ELEMENT OF A SET*)                      58;
0068:  (**) STRGFRL   =   (*NUMBER OF CHARACTERS IN A STRING FRAGMENT*)      10;
0069:  (**) NILVAL    =   (* = ORD(NIL) *)                                   0;
0070:  (**) CODEMAX   =   (*LENGTH OF THE CODE BUFFER*)                    1000;
0071:  (**) MXINT     =   (*LARGEST INTEGER VALUE*)                       32767;
0072:  (**) MAX10     =   (* = MAXINT DIV 10*)                             3276;
0073:  (**) MAXR10    =   (* 1/10 LARGEST REAL VALUE*)                   1.0E10;
0074:  (**) LOGMAX    =   (* LOG OF LARGEST REAL VALUE*)                     11;
0075:  (**) LOGMIN    =   (* LOG OF SMALLEST POSITIVE REAL VALUE*)          -11;
0076:  (**) LCPROGINIT  = (*INITIAL VALUE OF LOCATION COUNTER IN MAIN PROGR*) 0;
0077:  (**) ICSTART   =   (*INITIAL VALUE OF INSTR COUNTER IN MAIN PROGR*)    0;
0078:
0079:  (**) (*.GLOBCONST. DECLARATION OF GLOBAL CONSTANTS*)
0080:
0081:
0082:
0083:  TYPE                          (*DESCRIBING:*)
0084:                                (*************)
0085:     TRICKY = INTEGER;          (* USED TO AVOID EMPTY RECORDS *)
0086:     ALFA = PACKED ARRAY[1..ALFALENG] OF CHAR;
0087:
0088:
0089:                                 (*BASIC SYMBOLS*)
0090:                                 (***************)
0091:
0092:     SYMBOL = (IDENT,INTCONST,REALCONST,CHARCONST,STRINGCONST,NOTSY,MULOP,
0093:          ADDOP,RELOP,LPARENT,RPARENT,LBRACK,RBRACK,COMMA,SEMICOLON,
0094:          PERIOD,ARROW,COLON,BECOMES,LABELSY,CONSTSY,TYPESY,VARSY,
0095:          FUNCTSY,PROCSY,SETSY,PACKEDSY,ARRAYSY,RECORDSY,FILESY,
0096:          BEGINSY,IFSY,CASESY,REPEATSY,WHILESY,FORSY,WITHSY,
0097:          GOTOSY,ENDSY,ELSESY,UNTILSY,OFSY,DOSY,TOSY,DOWNTOSY,
0098:          THENSY,PROGRAMSY,OTHERSY);
0099:     OPERATOR = (MUL,RDIV,ANDOP,IDIV,IMOD,PLUS,MINUS,OROP,LTOP,LEOP,GEOP,
0100:           GTOP,NEOP,EQOP,INOP,NOOP);
0101:     SETOFSYS = SET OF SYMBOL;
0102:
0103:
0104:                                 (*CODE BUFFER LOCATIONS*)
0105:                                 (***********************)
0106:
0107:  (**)CODERANGE = (*.CODERANGE. DESCRIBES THE ADDRESS OF AN INSTRUCTION IN CODE
0108:                           BUFFER*) INTEGER;
0109:  (**)USFREF = (*.USFREF. DESCRIBES THE LOCATION OF AN UNSATISFIED REFERENCE.
0110:                 IT IS USED FOR:
0111:                 - FORWARD JUMPS
0112:                 - INSERTIONS OF BOUNDS OF CASE VARIABLES
0113:                 - INSERTIONS OF #LCMAX# FOR THE RUNTIME OVERFLOW TEST
0114:                 - IT CAN BE USED BY THE REAR END TO DELAY GENERATION OF CONSTANTS
0115:                *) CODERANGE;
0116:
0117:                                 (*CONSTANTS*)
0118:                                 (************)
0119:
0120:     CSTCLASS = (INT,REEL,PSET,STRG);
0121:     LOCOFREF = 'LOCREC;
0122:     CTAILP = ' CSTTAILREC;
0123:     STFRTYPE = PACKED ARRAY[1..STRGFRL] OF CHAR;
0124:     CSTTAILREC = RECORD NXTCSP: CTAILP;
0125:                    STFR: STFRTYPE;
0126:                  END;
0127:     STRDESCR = RECORD STRLEN: INTEGER; STRP: CTAILP END;
0128:
0129:     LOCREC = PACKED RECORD NXTREF: LOCOFREF;
0130:                LOC: USFREF;
0131:              END;
0132:
```

```
0133:     VALU = RECORD CASE CKIND: CSTCLASS OF (*CKIND NEVER SET NOR TESTED*)
0134:             INT: (IVAL: INTEGER);
0135:             REEL: (RVAL: REAL);
0136:             PSET: (PVAL: SET OF SETMIN..SETMAX);
0137:             STRG: (SVAL: STRDESCR)
0138:            END;
0139:
0140:                                 (*DATA STRUCTURES*)
0141:                                 (*****************)
0142:     LEVRANGE = 0..MAXLEVEL; ADDRRANGE = -MAXADDR..MAXADDR;
0143:  (**)PADDRRANGE = (*.PADDRRANGE. DESCRIBES AN ADDRRESS OF A FIELD IN A PACKED
0144:                    STRUCTURE*) ADDRRANGE;
0145:     STRUCTFORM = (SCALAR,SUBRANGE,POINTER,POWER,ARRAYS,RECORDS,FILES,
0146:            TAGFIELD,VARIANT);
0147:     DECLKIND = (STANDARD,DECLARED);
0148:  (**)SIZETYPE = (*.SIZETYPE. DESCRIBES THE SIZE OF A STRUCTURE IN TERMS OF WORDS,
0149:                  BYTES, BITS; IT CONTAINS A POSSIBLE ALIGNEMENT CONDITION.
0150:                  *) PACKED RECORD T : TRICKY END;
0151:     STP = ' STRUCTREC; CTP = ' IDENTREC;
0152:
0153:     STRUCTREC = PACKED RECORD
0154:            MARKED : BOOLEAN; (*FOR TEST PHASE ONLY*)
0155:            FTYPE: BOOLEAN; (* TRUE IFF THE STRUCTURE CONTAINS OR IS A FILE *)
0156:            SIZE: SIZETYPE;
0157:            (* IF FORM = RECORDS:
0158:               #SIZE# GIVES THE MAXIMUM SIZE, I.E. FOR THE CASE OF LONGEST
0159:                VARIANT PART.
0160:               IF FORM = TAGFIELD:
0161:               #SIZE# GIVES THE SIZE OF THE RECORD UP TO THE TAGFIELD, I.E.
0162:               WITHOUT ANY VARIANT PART.
0163:               IF FORM = VARIANT:
0164:               #SIZE# GIVES THE SIZE OF THE WHOLE RECORD IN THE CASE THE
0165:               TAGFIELD HAS THE VALUE #VARVAL#. *)
0166:            CASE FORM: STRUCTFORM OF
0167:             SCALAR:   (CASE SCALKIND: DECLKIND OF
0168:                         DECLARED: (FCONST: CTP));
0169:             SUBRANGE: (RANGETYPE: STP; MIN,MAX: VALU);
0170:             POINTER:  (ELTYPE: STP);
0171:             POWER:    (PCKDSET: BOOLEAN; ELSET: STP;
0172:  (**)                   (*.POWER. ADDITIONAL INFO*) );
0173:             ARRAYS:   (PCKDARR: BOOLEAN; AELTYPE,INXTYPE: STP;
0174:  (**)                   (*.ARRAYS. ADDITIONAL INFO*) );
0175:             RECORDS:  (PCKDREC: BOOLEAN; FIELDS,FSTFLD: CTP;
0176:                         RECVAR: STP);
0177:             FILES:    (PCKDFIL,TEXTFILE: BOOLEAN; FILTYPE: STP;
0178:  (**)                   (*.FILES. ADDITIONAL INFO*) );
0179:             TAGFIELD: (TGFLDP: CTP; FSTVAR: STP);
0180:             VARIANT:  (FSTVARFLD: CTP; NXTVAR,SUBVAR: STP;
0181:                         VARVAL: VALU)
0182:            END;
0183:
0184:
0185:                                 (*LABELS*)
0186:                                 (********)
0187:     LBP = 'LABREC;
0188:     GLABRG = 0..NGLAB;
0189:     GLABSET = SET OF GLABRG;
0190:     LABREC = PACKED RECORD
0191:                LABVAL: INTEGER; NEXTLAB: LBP;
0192:                LABLEV: LEVRANGE; LCNT: GLABRG; LABCONTR: INTEGER;
0193:                CASE DEFINED: BOOLEAN OF
0194:                 TRUE: (LABADDR: ADDRRANGE);
0195:                 FALSE: (FSTOCC: LOCOFREF)
0196:              END;
0197:
0198:
```

```
0199:                                (*NAMES*)
0200:                                (*******)
0201:
0202:     IDCLASS = (TYPES,KONST,VARS,FIELD,PROC,FUNC);
0203:     SETOFIDS = SET OF IDCLASS;
0204:     IDKIND = (ACTUAL,FORMAL);
0205:     DRCTINDRCT = (DRCT,INDRCT);
0206:     PFFORM = (DEF,NOTDEF,FORWDECL,EXTDECL,FTNDECL);
0207:
0208:     IDENTREC = PACKED RECORD
0209:            NAME: ALFA; LLINK: CTP; RLINK: CTP;
0210:            IDTYPE: STP; NEXT: CTP;
0211:            CASE KLASS: IDCLASS OF
0212:             KONST: (VALUES: VALU);
0213:             VARS:  (VKIND: DRCTINDRCT; VLEV: LEVRANGE;
0214:                     VADDR,PARADDR: ADDRRANGE);
0215:             FIELD: (CASE PCKDFLD: BOOLEAN OF
0216:                       FALSE: (FLDADDR: ADDRRANGE);
0217:                       TRUE:  (PFLDADDR: PADDRRANGE));
0218:             PROC,
0219:             FUNC:  (CASE PFDECKIND: DECLKIND OF
0220:                     STANDARD: (KEY: 1..NRSTDNAMES);
0221:                     DECLARED: (PFLEV: LEVRANGE; LCSAVE,PFADDR: ADDRRANGE;
0222:                                CASE PFKIND: IDKIND OF
0223:                                 ACTUAL: ( PFCNT : INTEGER; (*THIS NUMBER
0224:                                           IS SPECIFIC TO EACH PROC/FUNC
0225:                                           AND CAN BE USED TO FORM A SPE-
0226:                                           CIFIC IDENTIFIER FOR EACH PROC/
0227:                                           FUNC *)
0228:                                          FCTVALADDR: ADDRRANGE;
0229:                                          EXITLAB: GLABSET;
0230:  (**)                                    (*.PROCFUNC. ADDITIONAL INFORMATION*)
0231:                                          CASE PFDECL: PFFORM OF
0232:  (**)                                     (*.PROCFUNC. ADDITIONAL INFORMATION
0233:                                            FOR THE DIFFERENT CASES.
0234:                                            *) NOTDEF: (T: TRICKY)
0235:                    )          )         )
0236:            END;
0237:
0238:
0239:
0240:     EXTFILEP = ' FILEREC;
0241:     FILEREC = PACKED RECORD
0242:                 FILENAME: ALFA; FILEID: CTP;
0243:                 NXTP: EXTFILEP;
0244:                 DECLARED: BOOLEAN
0245:               END;
0246:
0247:
0248:     DISPRANGE = 0..DISPLIMIT;
0249:     WHERE = (BLCK,REC);
0250:
0251:
0252:                                (*MISCELLANEOUS*)
0253:                                (**************)
0254:
0255:
0256:    MARKP = ' BOOLEAN;        (*MARK AND RELEASE*)
0257:
0258:                (*TO DESCRIBE EXPRESSION CURRENTLY COMPILED*)
0259:                (****************************************)
0260:
0261:        ATTRKIND = (CST,VARBL);
0262:        ATTRP = ' ATTR; FORP = ' FORREC;
0263:
0264:        ATTR = RECORD FOLLOW,WOLLOF: ATTRP; SEQNR: INTEGER;
```

```
0265:                TYPTR: STP;
0266:  (**)          (*.ATTR. MACHINE DEPENDENT FIELDS*)
0267:                CASE KIND: ATTRKIND OF
0268:                 CST:   (CVAL: VALU;
0269:  (**)                    (*.CST. MACHINE DEPENDENT INFO*)
0270:                         );
0271:  (**)           VARBL: ( (*.VARBL. ADDRESS AND ACCESS INFO*)
0272:                          CASE EXPR: BOOLEAN OF
0273:                           FALSE: (FORPOINTER: FORP;
0274:                                   INPCKDSTR: BOOLEAN (*TRUE IFF PART OF A PACKED
0275:                                                       STRUCTURE*);
0276:  (**)                             (*.NOTEXPR. ADDRESS AND ACCESS INFO*)T1: TRICKY
0277:                                  );
0278:  (**)                     TRUE:  ( (*.EXPR. ADDRESS AND ACCESS INFO*)T2: TRICKY
0279:                         )        )
0280:           END;
0281:
0282:       FORREC = RECORD (*CURRENT FOR LOOPS*)
0283:                 NEXTFORP: FORP;
0284:                 CONTROLID: CTP;
0285:                END;
0286:
0287:  (*-----------------------------------------------------------------*)
0288:
0289:
0290:  VAR
0291:                   (*RETURNED BY SOURCE PROGRAM SCANNER
0292:                    INSYMBOL:
0293:                    **********)
0294:
0295:    SY: SYMBOL;                     (*LAST SYMBOL*)
0296:    OP: OPERATOR;                   (*CLASSIFICATION OF LAST SYMBOL*)
0297:    IVAL: INTEGER;                  (*VALUE OF LAST INTEGER CONSTANT*)
0298:    RVAL: REAL;                     (*VALUE OF LAST REAL CONSTANT*)
0299:    SVAL: STRDESCR;                 (*VALUE OF LAST STRING*)
0300:    LGTH: INTEGER;                  (*LENGTH OF LAST STRING CONSTANT*)
0301:    ID: ALFA;                       (*LAST IDENT (POSSIBLY TRUNCATED)*)
0302:    KK: 1..ALFALENG;                (*NR OF CHARS IN LAST IDENTIFIER*)
0303:    CH: CHAR;                       (*LAST CHARACTER*)
0304:    EOL: BOOLEAN;                   (*END OF LINE CONDITION*)
0305:
0306:
0307:                   (*COUNTERS:*)
0308:                   (***********)
0309:
0310:    CHCNT: INTEGER;                 (*CHARACTER COUNTER*)
0311:    LC,IC: ADDRRANGE;               (*DATA LOCATION AND INSTR COUNTER*)
0312:    LABCNT: INTEGER;                (*NUMBER OF EXTERNAL LABELS*)
0313:    EXTFILS: INTEGER;               (*NUMBER OF EXTERNAL FILES*)
0314:    FILECNT: INTEGER;               (*NUMBER OF LOCAL FILES*)
0315:    PCNT: INTEGER;                  (*NUMBER OF PROCSY/FUNCTIONS*)
0316:
0317:
0318:                   (*SWITCHES:*)
0319:                   (***********)
0320:
0321:    DP,                             (*DECLARATION PART*)
0322:    PRTERR: BOOLEAN;                (*TO ALLOW FORWARD REFERENCES
0323:                                    BY SUPPRESSING ERROR MESSAGE*)
0324:    DEBUG,LISTON,PMD,P: BOOLEAN;
0325:    TERMINATED : BOOLEAN;
0326:
0327:
0328:                   (*POINTERS:*)
0329:                   (***********)
0330:    INTPTR,REALPTR,CHARPTR,
```

```
0331:    BOOLPTR,NILPTR,TEXTPTR: STP;    (*POINTERS TO ENTRIES OF STD IDS*)
0332:    UTYPPTR,UCSTPTR,UVARPTR,
0333:    UFLDPTR,UPRCPTR,UFCTPTR,        (*POINTERS TO ENTRIES FOR UNDECL IDS*)
0334:    INPUTPTR,OUTPUTPTR,             (*ENTRIES FOR INPUT AND OUTPUT*)
0335:    FWPTR: CTP;                     (*HEAD OF CHAIN OF FORW  TYPE IDS*)
0336:    FSTLABP : LBP;                  (*HEAD OF LABEL CHAIN*)
0337:    FEXFILP: EXTFILEP;              (*HEAD OF LIST OF EXTERNAL FILES*)
0338:    GATTRP,BATTRP,ATTRHEAD: ATTRP;  (*EXPRESSION DESCRIPTOR,BOTTOM OF EXPRESSION
0339:                                      STACK, HEAD OF FREE DESCRIPTORS*)
0340:
0341:
0342:                    (*BOOKKEEPING OF DECLARATION LEVELS:*)
0343:                    (***********************************)
0344:
0345:    LEVEL: LEVRANGE;                (*CURRENT STATIC LEVEL*)
0346:    DISX,                           (*LEVEL OF LAST ID SRCHD BY SEARCHID*)
0347:    TOP: DISPRANGE;                 (*TOP OF DISPLAY*)
0348:
0349:    DISPLAY:                        (*WHERE:   MEANS:*)
0350:     ARRAY [DISPRANGE] OF
0351:      PACKED RECORD                  (*=BLCK:   ID IS VARIABLE ID*)
0352:       FNAME: CTP;                  (*=REC:   ID IS FIELD ID IN RECORD*)
0353:       CASE OCCUR: WHERE OF
0354:        REC: (WITHATTRP: ATTRP);
0355:        BLCK: (PFNAME: CTP);
0356:      END;
0357:
0358:                    (*ERROR MESSAGES:*)
0359:                    (*****************)
0360:
0361:    ERRINX: 0..10;                  (*NR OF ERRORS IN CURR SOURCE LINE*)
0362:    ERRORS: BOOLEAN;                (*TRUE IFF THE PROGRAM CONTAINS AN ERROR*)
0363:    ERRLIST:
0364:     ARRAY [1..10] OF
0365:      PACKED RECORD POS: 1..MAXCHCNT;
0366:             NMR: 1..400
0367:          END;
0368:
0369:
0370:                    (*MISCELLANEOUS*)
0371:                    (**************)
0372:
0373:    PROGNAME: ALFA;
0374:  (**) CODE : (*.CODE. CODE BUFFER*) ARRAY[0..CODEMAX] OF INTEGER;
0375:       CIX: CODERANGE; (*POINTER IN THE OUTPUT BUFFER OF THE CODE FILE.
0376:                         IT POINTS TO THE PLACE WHERE THE NEXT INSTRUCTION
0377:                         IS TO BE PUT IN.*)
0378:
0379:
0380:
0381:
0382:
0383:
0384:                    (*STRUCTURED CONSTANTS:*)
0385:                    (************************)
0386:
0387:    CONSTBEGSYS,SIMPTYPEBEGSYS,TYPEBEGSYS,BLOCKBEGSYS,SELECTSYS,FACBEGSYS,
0388:    STATBEGSYS,TYPEDELS: SETOFSYS;
0389:    RW: ARRAY [1..RESWORDS] OF ALFA;
0390:    LRW: ARRAY [0..ALFALENG] OF 0..RESWORDS;
0391:    RSY: ARRAY [1..RESWORDS] OF SYMBOL;
0392:    SSY: ARRAY [CHAR] OF SYMBOL;
0393:    ROP: ARRAY [1..RESWORDS] OF OPERATOR;
0394:    SOP: ARRAY [CHAR] OF OPERATOR;
0395:    NA:  ARRAY [1..NRSTDNAMES] OF ALFA;
0396:    BLANKSTRING: STFRTYPE;
```

```
0397:
0398:
0399:                     (*OUTPUT BUFFER:*)
0400:                     (****************)
0401:
0402:     LINE,SKIPLINE : ARRAY[1..MAXCHCNT] OF CHAR;
0403:     LINECOUNT : INTEGER;
0404:     SKIPPING,SKIPINLINE : BOOLEAN;
0405:     LCORIC : INTEGER; (*INDICATES THE VALUE OF #LC# OR #IC# AT THE
0406:                        BEGINNING OF THE CURRENT LINE *)
0407:     LASTCHCNT : INTEGER; (*GIVES THE POSITION OF THE END OF THE LAST SYMBOL*)
0408:
0409:  (**) (*.GLOBDECL. DECLARATION OF GLOBAL VARIABLES AND PROCEDURES*)
0410:
0411:
0412:   PROCEDURE MARK(VAR FP : MARKP);
0413:   BEGIN NEW(FP)
0414:   END (*MARK*);
0415:
0416:   PROCEDURE STOP;
0417:   BEGIN
0418:  (**) (*.STOP. STOP THE EXECUTION OF THE COMPILER*)
0419:     WRITELN(OUTPUT); WRITE(OUTPUT,# ---STOP---#); WRITELN(OUTPUT);
0420:   END (*STOP*);
0421:
0422:   PROCEDURE WRITEERRORS;
0423:    VAR LASTPOS,FREEPOS,CURRPOS,CURRNMR,F,K: INTEGER;
0424:   BEGIN WRITE(OUTPUT,# *******#,# #:NSP);
0425:    LASTPOS := 0; FREEPOS := 1;
0426:    FOR K := 1 TO ERRINX DO
0427:     BEGIN
0428:      WITH ERRLIST[K] DO
0429:       BEGIN CURRPOS := POS; CURRNMR := NMR END;
0430:      IF CURRPOS = LASTPOS THEN WRITE(OUTPUT,#,#)
0431:      ELSE
0432:       BEGIN
0433:        IF FREEPOS > CURRPOS THEN
0434:         BEGIN WRITELN(OUTPUT); WRITE(OUTPUT,# *******#,# #:NSP);
0435:          FREEPOS := 1;
0436:         END;
0437:        WHILE FREEPOS < CURRPOS DO
0438:         BEGIN WRITE(OUTPUT,# #); FREEPOS := FREEPOS + 1 END;
0439:        WRITE(OUTPUT,#'#);
0440:         LASTPOS := CURRPOS
0441:       END;
0442:      IF CURRNMR < 10 THEN F := 1
0443:      ELSE IF CURRNMR < 100 THEN F := 2
0444:       ELSE F := 3;
0445:      WRITE(OUTPUT,CURRNMR:F);
0446:      FREEPOS := FREEPOS + F + 1
0447:     END;
0448:    WRITELN(OUTPUT); ERRINX := 0
0449:   END (*WRITEERRORS*) ;
0450:
0451:     PROCEDURE WRCI(N:INTEGER);
0452:  (**) (*.WRCI. WRITE AN INTEGER CONSTANT ON FILE OUTPUT. THIS PROCEDURE
0453:           IS USED IN #PRINTTABLES# AND #PRINTATTR# TO PRINT CONTROL OUTPUT
0454:         *) BEGIN WRITE(OUTPUT,N:9);
0455:     END (*WRCI*);
0456:
0457:     PROCEDURE WRCR(R:REAL);
0458:  (**) (*.WRCR. IDEM FOR A REAL CONSTANT*) BEGIN WRITE(OUTPUT,R:10);
0459:     END (*WRCR*);
0460:
0461:     PROCEDURE WRN(N:INTEGER);
0462:  (**) (*.WRN. IDEM FOR AN ADDRESS*) BEGIN WRITE(OUTPUT,N:6);
```

```
0463:    END (*WRN*);
0464:
0465:   PROCEDURE WRPN(PN: ADDRRANGE);
0466:  (**) (*.WRPN. IDEM FOR A PACKED ADDRESS*) BEGIN
0467:   END (*WRPN*);
0468:
0469:    PROCEDURE WRSP;
0470:    BEGIN WRITE(OUTPUT,# #:4);
0471:    END (*WRSP*);
0472:
0473:   PROCEDURE ENDOFLINE;
0474:    VAR I : INTEGER;
0475:    BEGIN LINECOUNT := LINECOUNT + 1;
0476:     IF LISTON OR (ERRINX>0) OR SKIPINLINE THEN
0477:      BEGIN
0478:       WRCI(LINECOUNT); WRSP; WRN(LCORIC); WRSP;
0479:       FOR I := 1 TO CHCNT DO WRITE(OUTPUT,LINE[I]);
0480:       WRITELN(OUTPUT);
0481:       IF ERRINX>0 THEN WRITEERRORS;
0482:       IF SKIPINLINE THEN
0483:        BEGIN WRITE(OUTPUT,# **SKIP*#,# #:NSP);
0484:         FOR I := 1 TO CHCNT DO
0485:          BEGIN WRITE(OUTPUT,SKIPLINE[I]);
0486:           SKIPLINE[I] := # #
0487:          END;
0488:         WRITELN(OUTPUT)
0489:        END
0490:      END;
0491:     IF NOT TERMINATED THEN
0492:      IF EOF(INPUT) THEN
0493:       BEGIN WRITELN(OUTPUT,# EOF ENCOUNTERED#); ERRORS := TRUE; STOP;
0494:       END;
0495:     CHCNT := 0; LASTCHCNT := 1; SKIPINLINE := FALSE;
0496:     IF DP THEN LCORIC := LC ELSE LCORIC := IC
0497:   END (*ENDOFLINE*) ;
0498:
0499:   PROCEDURE ERROR(FERRNR: INTEGER);
0500:   BEGIN ERRORS := TRUE;
0501:    IF ERRINX >= 9 THEN
0502:     BEGIN ERRLIST[10].NMR := 255; ERRINX := 10 END
0503:    ELSE
0504:     BEGIN ERRINX := ERRINX + 1;
0505:      ERRLIST[ERRINX].NMR := FERRNR
0506:     END;
0507:    ERRLIST[ERRINX].POS := CHCNT
0508:   END (*ERROR*) ;
0509:
0510:   PROCEDURE INSYMBOL;
0511:    (*READ NEXT BASIC SYMBOL OF SOURCE PROGRAM AND RETURN ITS DESCRIPTION
0512:    IN THE GLOBAL VARIABLES SY, OP, ID, IVAL, RVAL, SVAL AND LGTH*)
0513:    LABEL 1,2;
0514:    VAR I,K,SCALE,EXP: INTEGER;
0515:      R,FAC,ESTIM: REAL; SIGN: BOOLEAN; LSF: STFRTYPE;
0516:      DIGIT: ARRAY [1..DIGMAX] OF 0..9;
0517:      APO,STRINGEND: BOOLEAN; NXTP,TAILP,LCSP: CTAILP;
0518:      SAVE : CHAR; REALFLAG : BOOLEAN;
0519:  (**) (*.INSYMBOL. OTHER LOCAL DECLARATIONS*)
0520:
0521:    PROCEDURE NEXTCH;
0522:   VAR ERRFLAG: BOOLEAN;
0523:    BEGIN
0524:     EOL := EOLN(INPUT);
0525:     READ(INPUT,CH); (*NOTE THAT IF EOL = TRUE, #READ# MUST DELIVER A BLANK*)
0526:     CHCNT := CHCNT + 1; LINE[CHCNT] := CH;
0527:     IF CHCNT = MAXCHCNT THEN
0528:      IF NOT EOL THEN
```

```
0529:       BEGIN ERRFLAG := CH <> # #;
0530:        REPEAT EOL := EOLN(INPUT); READ(INPUT,CH);
0531:         ERRFLAG := ERRFLAG OR (CH <> # #);
0532:        UNTIL EOL;
0533:        IF ERRFLAG THEN ERROR(180)
0534:       END
0535:    END (*NEXTCH*) ;
0536:
0537:    FUNCTION DIGVAL(CH: CHAR): INTEGER;
0538:  (**) (*.DIGVAL. RETURNS THE INTEGER VALUE OF A DIGIT:
0539:             #0# GIVES 0, #1# GIVES 1, ... *) BEGIN
0540:    END (*DIGVAL*);
0541:
0542:    FUNCTION LETTERS(CH: CHAR): BOOLEAN;
0543:  (**) (*.LETTERS. RETURN TRUE IFF #CH# IS A LETTER. THIS PROCEDURE
0544:          CAN BE REMOVED IF THE CALLS ARE REPLACED BY INLINE
0545:          CODE *) BEGIN
0546:    END (*LETTERS*);
0547:
0548:    FUNCTION DIGITS(CH: CHAR): BOOLEAN;
0549:  (**) (*.DIGITS. RETURN TRUE IFF #CH# IS A DIGIT. THIS PROCEDURE
0550:          CAN BE REMOVED IF THE CALLS ARE REPLACED BY INLINE
0551:           CODE *) BEGIN
0552:    END (*DIGITS*);
0553:
0554:    FUNCTION LEGAL(CH : CHAR) : BOOLEAN;
0555:  (**) (*.LEGAL. RETURN TRUE IFF #CH# IS A LEGAL CHARACTER. THIS
0556:          PROCEDURE CAN BE REMOVED IF THE CALLS ARE REPLACED BY
0557:          INLINE CODE *) BEGIN
0558:    END (*LEGAL*);
0559:
0560:    PROCEDURE OPTIONS;
0561:     VAR  CH1: CHAR;
0562:     BEGIN
0563:     REPEAT NEXTCH;
0564:     IF (CH=#A#) OR (CH=#L#) OR (CH=#P#) OR (CH=#T#) THEN
0565:       BEGIN CH1 := CH; NEXTCH;
0566:        CASE CH1 OF
0567:        #A#: IF (CH=#+#) OR (CH=#-#) THEN P := CH=#+#; (*TEST PHASE ONLY*)
0568:        #L#: IF (CH=#+#) OR (CH=#-#) THEN LISTON := CH=#+#;
0569:        #P#: IF (CH=#+#) OR (CH=#-#) THEN PMD := CH=#+#;
0570:        #T#: IF (CH=#+#) OR (CH=#-#) THEN DEBUG := CH=#+#
0571:        END;
0572:       END
0573:      ELSE NEXTCH;
0574:      NEXTCH;
0575:     UNTIL CH <> #,#
0576:    END (*OPTIONS*);
0577:
0578:   BEGIN (*INSYMBOL*)
0579:   1:
0580:    IF SKIPPING THEN
0581:     BEGIN FOR I := LASTCHCNT TO CHCNT-1 DO
0582:      BEGIN SKIPLINE[I] := #-#; SKIPINLINE := TRUE END
0583:     END;
0584:    LASTCHCNT := CHCNT;
0585:    (*LOOP UNTIL EOL:*)
0586:    WHILE (CH = # #)AND NOT EOL DO NEXTCH;
0587:    WHILE EOL DO
0588:     BEGIN ENDOFLINE; NEXTCH;
0589:      WHILE (CH = # #)AND NOT EOL DO NEXTCH
0590:     END;
0591:    SY := OTHERSY;
0592:  (**) (*.INSYMBOL. INSERT HERE A PIECE OF CODE TO TEST IF THE CHARACTER
0593:          #CH# IS A LEGAL ONE. *) IF LEGAL(CH) THEN
0594:    CASE CH OF
```

```
0595:     #A#,#B#,#C#,#D#,#E#,#F#,#G#,#H#,#I#,
0596:     #J#,#K#,#L#,#M#,#N#,#O#,#P#,#Q#,#R#,
0597:     #S#,#T#,#U#,#V#,#W#,#X#,#Y#,#Z#:
0598:      BEGIN K := 0;
0599:       REPEAT
0600:        IF K < ALFALENG THEN
0601:         BEGIN K := K + 1; ID[K] := CH END;
0602:        NEXTCH
0603:  (**) UNTIL (*.INSYMBOL. INSERT HERE A CONDITION WHICH IS FALSE IFF
0604:                #CH# IS A LETTER OR A DIGIT.
0605:              *) NOT (LETTERS(CH) OR DIGITS(CH));
0606:       IF K >= KK THEN KK := K
0607:       ELSE
0608:        REPEAT ID[KK] := # #; KK := KK - 1;
0609:        UNTIL KK = K;
0610:       FOR I := LRW[K-1] + 1 TO LRW[K] DO
0611:        IF RW[I] = ID THEN
0612:         BEGIN SY := RSY[I]; OP := ROP[I]; GOTO 2 END;
0613:       SY := IDENT; OP := NOOP;
0614:   2:    END;
0615:     #0#,#1#,#2#,#3#,#4#,#5#,#6#,#7#,#8#,#9#:
0616:      BEGIN SY := INTCONST; OP := NOOP;
0617:       I := 0;
0618:      REALFLAG := FALSE;
0619:       REPEAT I := I + 1;
0620:        IF I <= DIGMAX THEN DIGIT[I] :=
0621:  (**)    (*.INSYMBOL. INSERT HERE A PIECE OF CODE TO GIVE A VALUE OF THE DIGIT
0622:            #CH#*) DIGVAL(CH)
0623:        ELSE
0624:         BEGIN ERROR(203); I := 1 END;
0625:        NEXTCH
0626:  (**) UNTIL (*.INSYMBOL. INSERT HERE A CONDITION WHICH IS FALSE IFF
0627:                #CH# IS A DIGIT*) NOT DIGITS(CH);
0628:       IVAL := 0;
0629:  (*HOOK*) (*EVALUATE OCTAL OR HEXADECIMAL NUMBER HERE*)
0630:         FOR K := 1 TO I DO
0631:          IF IVAL <= MAX10 THEN
0632:           IVAL := 10*IVAL + DIGIT[K]
0633:          ELSE BEGIN ERROR(203); IVAL := 0 END;
0634:         SCALE := 0;
0635:         IF CH = #.# THEN
0636:          BEGIN NEXTCH;
0637:           IF CH = #.# THEN CH := #:#
0638:           ELSE
0639:            BEGIN RVAL := IVAL; SY := REALCONST;
0640:             REALFLAG := TRUE;
0641:  (**)       IF (*.INSYMBOL. INSERT HERE A CONDITION WHICH IS FALSE
0642:                   IFF #CH# IS A DIGIT*) NOT DIGITS(CH)
0643:              THEN ERROR(201)
0644:             ELSE
0645:              REPEAT
0646:               IF RVAL <= MAXR10 THEN RVAL := 10.0*RVAL +
0647:  (**)          (*.INSYMBOL. INSERT HERE A PIECE OF CODE TO GIVE THE VALUE OF THE
0648:                  DIGIT #CH# *) DIGVAL(CH)
0649:               ELSE BEGIN ERROR(205); RVAL := 0.0 END;
0650:               SCALE := SCALE - 1; NEXTCH
0651:  (**)         UNTIL (*.INSYMBOL. INSERT HERE A CONDITION WHICH IS
0652:                     FALSE IFF #CH# IS A DIGIT*) NOT DIGITS(CH);
0653:            END
0654:          END;
0655:         IF CH = #E# THEN
0656:          BEGIN
0657:           REALFLAG := TRUE;
0658:           IF SCALE = 0 THEN
0659:            BEGIN RVAL := IVAL; SY := REALCONST END;
0660:           SIGN := FALSE; NEXTCH;
```

```
0661:           IF CH = #+# THEN NEXTCH
0662:           ELSE
0663:            IF CH = #-# THEN
0664:             BEGIN SIGN := TRUE; NEXTCH END;
0665:           EXP := 0;
0666:  (**)     IF (*.INSYMBOL. INSERT HERE A CONDITION WHICH IS FALSE
0667:                IFF #CH# IS A DIGIT*) NOT DIGITS(CH)
0668:            THEN ERROR(201)
0669:           ELSE
0670:            REPEAT
0671:             IF EXP <= MAX10 THEN EXP := 10*EXP +
0672:  (**)        (*.INSYMBOL. INSERT HERE A PIECE OF CODE TO GIVE THE VALUE OF THE
0673:                DIGIT #CH# *) DIGVAL(CH)
0674:             ELSE BEGIN ERROR(203); EXP := 0 END;
0675:             NEXTCH
0676:  (**)      UNTIL (*.INSYMBOL. INSERT HERE A CONDITION WHICH IS
0677:                  FALSE IFF #CH# IS A DIGIT*) NOT DIGITS(CH);
0678:           IF SIGN THEN SCALE := SCALE - EXP
0679:           ELSE SCALE := SCALE + EXP
0680:          END;
0681:       IF REALFLAG THEN
0682:       BEGIN
0683:         ESTIM := SCALE + I;
0684:         IF ESTIM > LOGMAX THEN ERROR(205)
0685:         ELSE
0686:          IF ESTIM < LOGMIN THEN ERROR(206)
0687:          ELSE
0688:           IF SCALE <> 0 THEN
0689:            BEGIN R := 1.0;
0690:             IF SCALE < 0 THEN
0691:              BEGIN FAC := 0.1; SCALE := -SCALE END
0692:             ELSE FAC := 10.0;
0693:             REPEAT IF ODD(SCALE) THEN R := R*FAC;
0694:              FAC := SQR(FAC); SCALE := SCALE DIV 2
0695:             UNTIL SCALE = 0;   (*NOW R = 10'SCALE*)
0696:             RVAL := RVAL*R
0697:            END;
0698:       END;
0699:      END;
0700:     ####
0701:         :
0702:      BEGIN OP := NOOP; APO := FALSE; STRINGEND := FALSE;
0703:       LGTH := 0; I := 0; LCSP := NIL; LSF := BLANKSTRING;
0704:       NEXTCH;
0705:       REPEAT
0706:        IF EOL THEN
0707:         BEGIN ERROR(202); STRINGEND := TRUE END
0708:        ELSE
0709:         IF (CH <> ####)OR APO THEN
0710:          BEGIN
0711:           IF I = STRGFRL THEN
0712:            BEGIN NEW(TAILP);
0713:             WITH TAILP' DO
0714:              BEGIN NXTCSP := LCSP; STFR := LSF END;
0715:             LCSP := TAILP; I := 0; LSF := BLANKSTRING;
0716:            END;
0717:           I := I + 1; LGTH := LGTH + 1; APO := FALSE;
0718:           LSF[I] := CH;
0719:           SAVE := CH;
0720:           NEXTCH;
0721:          END
0722:         ELSE
0723:           BEGIN APO := TRUE;
0724:           NEXTCH; STRINGEND := CH <> ####
0725:          END
0726:       UNTIL STRINGEND;
```

```
0727:       IF LGTH = 1 THEN
0728:  (**)  BEGIN SY := CHARCONST; IVAL := (*.ORDCHAR. MAP THE CHARACTER
0729:                       #SAVE# INTO ITS ORDINAL NUMBER (IN THE SYSTEM OF THE
0730:                       GOAL COMPUTER). IF THE HOST AND THE GOAL MACHINE ARE
0731:                       THE SAME, USE #ORD(SAVE)# *) ORD(SAVE);
0732:        END
0733:       ELSE
0734:        BEGIN
0735:         NEW(TAILP);
0736:         WITH TAILP' DO
0737:          BEGIN NXTCSP := LCSP; STFR := LSF END;
0738:         (*REVERSE POINTERS:*)
0739:         LCSP := NIL;
0740:         WHILE TAILP <> NIL DO
0741:          WITH TAILP' DO
0742:           BEGIN NXTP := NXTCSP; NXTCSP := LCSP;
0743:            LCSP := TAILP; TAILP := NXTP
0744:           END;
0745:          WITH SVAL DO BEGIN STRLEN := LGTH; STRP := LCSP END;
0746:         SY := STRINGCONST
0747:        END
0748:      END;
0749:     #:#
0750:          :
0751:      BEGIN OP := NOOP; NEXTCH;
0752:       IF CH = #=# THEN
0753:        BEGIN SY := BECOMES; NEXTCH END
0754:       ELSE SY := COLON
0755:      END;
0756:     #.#:
0757:      BEGIN OP := NOOP; NEXTCH;
0758:       IF CH = #.# THEN
0759:        BEGIN SY := COLON; NEXTCH END
0760:       ELSE SY := PERIOD
0761:      END;
0762:     #(#:
0763:      BEGIN NEXTCH;
0764:       IF CH = #*# THEN
0765:        BEGIN NEXTCH;
0766:         IF CH = #$# THEN OPTIONS;
0767:         REPEAT
0768:          (*LOOP UNTIL CH = #*#:*)
0769:          IF EOL THEN ENDOFLINE;
0770:          WHILE CH <> #*# DO
0771:           BEGIN NEXTCH;
0772:            IF EOL THEN ENDOFLINE
0773:           END;
0774:          NEXTCH
0775:         UNTIL CH = #)#;
0776:         NEXTCH; GOTO 1
0777:        END;
0778:       SY := LPARENT; OP := NOOP
0779:      END;
0780:     #<#:
0781:      BEGIN NEXTCH; SY := RELOP;
0782:       IF CH = #=# THEN
0783:        BEGIN OP := LEOP; NEXTCH END
0784:       ELSE
0785:        IF CH = #>#  THEN
0786:         BEGIN OP := NEOP; NEXTCH END
0787:        ELSE OP := LTOP
0788:      END;
0789:     #>#:
0790:      BEGIN NEXTCH; SY := RELOP;
0791:       IF CH = #=# THEN
0792:        BEGIN OP := GEOP; NEXTCH END
```

```
0793:       ELSE OP := GTOP
0794:      END;
0795:     #*#,#/#,#+#,#-#,#=#,#)#,
0796:     #[#,#]#,#,#,#;#,#'#
0797:                          :
0798:      BEGIN SY := SSY[CH]; OP := SOP[CH];
0799:       NEXTCH
0800:      END
0801:    END (*CASE*)
0802:  (**)(*.ILLCHAR. INSERT HERE A PIECE OF CODE FOR TREATMENT OF AN
0803:      ILLEGAL CHARACTER.*) ELSE BEGIN ERROR(22); NEXTCH END;
0804:   END (*INSYMBOL*) ;
0805:
0806:   PROCEDURE ENTERID(FCP: CTP);
0807:    (*ENTER ID POINTED AT BY FCP INTO THE NAME-TABLE,
0808:     WHICH ON EACH DECLARATION LEVEL IS ORGANISED AS
0809:     AN UNBALANCED BINARY TREE*)
0810:    VAR NAM: ALFA; LCP, LCP1: CTP; LLEFT: BOOLEAN;
0811:   BEGIN NAM := FCP'.NAME;
0812:    LCP := DISPLAY[TOP].FNAME;
0813:    IF LCP = NIL THEN
0814:     DISPLAY[TOP].FNAME := FCP
0815:    ELSE
0816:     BEGIN
0817:      REPEAT LCP1 := LCP;
0818:       IF LCP'.NAME = NAM THEN   (*NAME CONFLICT, FOLLOW RIGHT LINK*)
0819:        BEGIN ERROR(101); LCP := LCP'.RLINK; LLEFT := FALSE END
0820:       ELSE
0821:        IF LCP'.NAME < NAM THEN
0822:         BEGIN LCP := LCP'.RLINK; LLEFT := FALSE END
0823:        ELSE BEGIN LCP := LCP'.LLINK; LLEFT := TRUE END
0824:      UNTIL LCP = NIL;
0825:      IF LLEFT THEN LCP1'.LLINK := FCP ELSE LCP1'.RLINK := FCP
0826:     END;
0827:    FCP'.LLINK := NIL; FCP'.RLINK := NIL
0828:   END (*ENTERID*) ;
0829:
0830:   PROCEDURE SEARCHSECTION(FCP: CTP; VAR FCP1: CTP);
0831:    (*TO FIND RECORD FIELDS AND FORWARD DECLARED PROCEDURE ID#S
0832:     --> PROCEDURE PROCDECLARATION
0833:     --> PROCEDURE SELECTOR*)
0834:    LABEL 1;
0835:   BEGIN
0836:    WHILE FCP <> NIL DO
0837:     IF FCP'.NAME = ID THEN GOTO 1
0838:     ELSE IF FCP'.NAME < ID THEN FCP := FCP'.RLINK
0839:      ELSE FCP := FCP'.LLINK;
0840:  1:  FCP1 := FCP
0841:   END (*SEARCHSECTION*) ;
0842:
0843:   PROCEDURE SEARCHID(FIDCLS: SETOFIDS; VAR FCP: CTP);
0844:    LABEL 1;
0845:    VAR LCP: CTP;
0846:   BEGIN
0847:    FOR DISX := TOP DOWNTO 0 DO
0848:     BEGIN LCP := DISPLAY[DISX].FNAME;
0849:      WHILE LCP <> NIL DO
0850:       IF LCP'.NAME = ID THEN
0851:        IF LCP'.KLASS IN FIDCLS THEN GOTO 1
0852:        ELSE
0853:         BEGIN IF PRTERR THEN ERROR(103);
0854:          LCP := LCP'.RLINK
0855:         END
0856:       ELSE
0857:        IF LCP'.NAME < ID THEN
0858:         LCP := LCP'.RLINK
```

```
0859:        ELSE LCP := LCP'.LLINK
0860:     END;
0861:    (*SEARCH NOT SUCCSESSFUL; SUPPRESS ERROR MESSAGE IN CASE
0862:     OF FORWARD REFERENCED TYPE ID IN POINTER TYPE DEFINITION
0863:     OR VARIANTS WITHOUT TAGFIELDS
0864:     --> PROCEDURE FIELDLIST
0865:     --> PROCEDURE SIMPLETYPE*)
0866:    IF PRTERR THEN
0867:     BEGIN ERROR(104);
0868:      (*TO AVOID RETURNING NIL, REFERENCE AN ENTRY
0869:       FOR AN UNDECLARED ID OF APPROPRIATE CLASS
0870:       --> PROCEDURE ENTERUNDECL*)
0871:      IF TYPES IN FIDCLS THEN LCP := UTYPPTR
0872:      ELSE
0873:       IF VARS IN FIDCLS THEN LCP := UVARPTR
0874:       ELSE
0875:        IF FIELD IN FIDCLS THEN LCP := UFLDPTR
0876:        ELSE
0877:         IF KONST IN FIDCLS THEN LCP := UCSTPTR
0878:         ELSE
0879:          IF PROC IN FIDCLS THEN LCP := UPRCPTR
0880:          ELSE LCP := UFCTPTR;
0881:     END;
0882:  1:  FCP := LCP
0883:   END (*SEARCHID*) ;
0884:
0885:
0886:   PROCEDURE GETBOUNDS(FSP: STP; VAR FMIN,FMAX: INTEGER);
0887:    (*GET INTERNAL BOUNDS OF SUBRANGE OR SCALAR TYPE*)
0888:    (*ASSUME (FSP <> INTPTR) AND (FSP <> REALPTR)*)
0889:    BEGIN
0890:     FMIN := 0; FMAX := 0;
0891:     IF FSP <> NIL THEN
0892:      WITH FSP' DO
0893:       IF FORM = SUBRANGE THEN
0894:        BEGIN FMIN := MIN.IVAL; FMAX := MAX.IVAL END
0895:       ELSE
0896:        BEGIN FMIN := 0; FMAX := 0;
0897:         IF FORM = SCALAR THEN
0898:          BEGIN
0899:           IF SCALKIND = STANDARD THEN
0900:            BEGIN IF FSP = CHARPTR THEN FMAX := ORDCHARMAX
0901:            END
0902:           ELSE
0903:            IF FSP'.FCONST <> NIL THEN
0904:             FMAX := FSP'.FCONST'.VALUES.IVAL
0905:          END
0906:        END
0907:    END (*GETBOUNDS*);
0908:
0909:   PROCEDURE SKIP(FSYS: SETOFSYS);
0910:    (*SKIP INPUT STRING UNTIL RELEVANT SYMBOL FOUND*)
0911:   BEGIN SKIPPING := TRUE; WHILE NOT (SY IN FSYS) DO INSYMBOL;
0912:    SKIPPING := FALSE
0913:   END (*SKIP*) ;
0914:
0915:
0916:   PROCEDURE BLOCK(FSYS: SETOFSYS; FSY: SYMBOL; FPROCP: CTP);
0917:    VAR LSY: SYMBOL; FLABP: LBP; LFORWCNT: INTEGER; LCHCNT: INTEGER;
0918:
0919:    PROCEDURE CHECKFORW(FCP: CTP);
0920:    (*PRINT ERROR MESSAGE FOR FORWARD DECLARED PROCEDURE*)
0921:    BEGIN
0922:     IF FCP <> NIL THEN
0923:      WITH FCP' DO
0924:       BEGIN
```

```
0925:        IF KLASS IN [PROC,FUNC] THEN
0926:         IF PFKIND = ACTUAL THEN
0927:          IF PFDECL = FORWDECL THEN
0928:           BEGIN ERROR(183);
0929:            WRITELN(OUTPUT,# *** UNDECLARED PROCEDURE:#, NAME);
0930:           END;
0931:        CHECKFORW(LLINK); CHECKFORW(RLINK);
0932:       END;
0933:    END (*CHECKFORW*);
0934:
0935:    PROCEDURE INITSIZE(VAR FSIZE : SIZETYPE);
0936:  (**) (*.INITSIZE. INITIALIZE #FSIZE# (TO AVOID AN UNDEFINED VALUE *) BEGIN
0937:    END (*INITSIZE*);
0938:
0939:    PROCEDURE SIZER(FSP: STP);
0940:  (**) (*.SIZER. SET THE SIZE OF #FSP#.
0941:           #SIZER# WILL NOT BE CALLED
0942:             IF FSP'.FORM IN [RECORDS,VARIANT,TAGFIELD] *) BEGIN
0943:    END (*SIZER*);
0944:
0945:    FUNCTION COMPTYPES(FSP1,FSP2: STP) : BOOLEAN;
0946:     (*DECIDE WHETHER STRUCT POINTED AT BY #FSP1# AND #FSP2# ARE COMPATIBLE*)
0947:     VAR NXT1,NXT2: CTP; COMP: BOOLEAN;
0948:
0949:     FUNCTION COMPLISTS(FCP1,FCP2: CTP; FSP1,FSP2: STP) : BOOLEAN;
0950:      (*DECIDE WHETHER FIELDLISTS ARE COMPATIBLE*)
0951:      (*FCP1, FCP2: HEADS OF FIELDLISTS; FSP1, FSP2: POINTERS TO HEAD OF
0952:       VARIANT CHAIN*)
0953:      VAR COMP: BOOLEAN; NXT1, NXT2: STP;
0954:     BEGIN COMP := TRUE;
0955:      WHILE COMP AND (FCP1 <> NIL)AND (FCP2 <> NIL) DO
0956:       BEGIN COMP := COMP AND (FCP1'.IDTYPE = FCP2'.IDTYPE);
0957:        FCP1 := FCP1'.NEXT; FCP2 := FCP2'.NEXT
0958:       END;
0959:      COMP := COMP AND (FCP1 = FCP2);
0960:      IF (FSP1 <> NIL)AND (FSP2 <> NIL) THEN
0961:       BEGIN
0962:        IF (FSP1'.TGFLDP <> NIL)AND (FSP2'.TGFLDP <> NIL) THEN
0963:         COMP := COMP AND (FSP1'.TGFLDP'.IDTYPE = FSP2'.TGFLDP'.IDTYPE);
0964:        NXT1 := FSP1'.FSTVAR; NXT2 := FSP2'.FSTVAR;
0965:        WHILE COMP AND (NXT1 <> NIL)AND (NXT2 <> NIL) DO
0966:         BEGIN COMP := COMP AND COMPLISTS(NXT1'.FSTVARFLD,NXT2'.FSTVARFLD,
0967:                                          NXT1'.SUBVAR,NXT2'.SUBVAR);
0968:          NXT1 := NXT1'.NXTVAR; NXT2 := NXT2'.NXTVAR
0969:         END;
0970:        COMPLISTS := COMP AND (NXT1 = NXT2)
0971:       END
0972:      ELSE COMPLISTS := COMP AND (FSP1 = FSP2)
0973:     END (*COMPLISTS*) ;
0974:
0975:     FUNCTION EQUALBOUNDS(FSP1,FSP2: STP) : BOOLEAN;
0976:      VAR LMIN1,LMIN2,LMAX1,LMAX2: INTEGER;
0977:     BEGIN GETBOUNDS(FSP1,LMIN1,LMAX1);
0978:      GETBOUNDS(FSP2,LMIN2,LMAX2);
0979:      EQUALBOUNDS := (LMIN1 = LMIN2)AND (LMAX1 = LMAX2)
0980:     END (*EQUALBOUNDS*) ;
0981:
0982:    BEGIN (*COMPTYPES*)
0983:     IF FSP1 = FSP2 THEN COMPTYPES := TRUE
0984:     ELSE
0985:      IF (FSP1 <> NIL)AND (FSP2 <> NIL) THEN
0986:       IF FSP1'.FORM = FSP2'.FORM THEN
0987:        CASE FSP1'.FORM OF
0988:         SCALAR:
0989:          IF (FSP1'.SCALKIND = STANDARD)OR(FSP2'.SCALKIND = STANDARD) THEN
0990:           COMPTYPES := FALSE
```

```
0991:          ELSE
0992:           BEGIN NXT1 := FSP1'.FCONST; NXT2 := FSP2'.FCONST;
0993:            COMP := TRUE;
0994:            WHILE COMP AND (NXT1 <> NIL)AND (NXT2 <> NIL) DO
0995:             BEGIN COMP := COMP AND (NXT1'.NAME = NXT2'.NAME);
0996:              NXT1 := NXT1'.NEXT; NXT2 := NXT2'.NEXT
0997:             END;
0998:            COMPTYPES := COMP AND (NXT1 = NXT2)
0999:           END;
1000:         SUBRANGE:
1001:          COMPTYPES := COMPTYPES(FSP1'.RANGETYPE,FSP2'.RANGETYPE);
1002:         POINTER:
1003:          COMPTYPES := COMPTYPES(FSP1'.ELTYPE,FSP2'.ELTYPE);
1004:         POWER:
1005:          COMPTYPES := (FSP1'.PCKDSET = FSP2'.PCKDSET) AND
1006:                       COMPTYPES(FSP1'.ELSET,FSP2'.ELSET);
1007:         ARRAYS:
1008:          BEGIN COMP := (FSP1'.PCKDARR = FSP2'.PCKDARR) AND
1009:                        COMPTYPES(FSP1'.INXTYPE,FSP2'.INXTYPE) AND
1010:                        (FSP1'.AELTYPE = FSP2'.AELTYPE);
1011:            COMPTYPES := COMP AND EQUALBOUNDS(FSP1'.INXTYPE,
1012:                         FSP2'.INXTYPE);
1013:          END;
1014:         RECORDS:
1015:          COMPTYPES := (FSP1'.PCKDREC = FSP2'.PCKDREC)
1016:                 AND COMPLISTS(FSP1'.FSTFLD,FSP2'.FSTFLD,
1017:                 FSP1'.RECVAR,FSP2'.RECVAR);
1018:         FILES:
1019:          COMPTYPES := (FSP1'.PCKDFIL = FSP2'.PCKDFIL)
1020:               AND (FSP1'.FILTYPE = FSP2'.FILTYPE)
1021:        END (*CASE*)
1022:       ELSE (*FSP1'.FORM <> FSP2'.FORM*)
1023:        IF FSP1'.FORM = SUBRANGE THEN
1024:         COMPTYPES := COMPTYPES(FSP1'.RANGETYPE,FSP2)
1025:        ELSE
1026:         IF FSP2'.FORM = SUBRANGE THEN
1027:          COMPTYPES := COMPTYPES(FSP1,FSP2'.RANGETYPE)
1028:         ELSE COMPTYPES := FALSE
1029:      ELSE COMPTYPES := TRUE
1030:    END (*COMPTYPES*) ;
1031:
1032:    FUNCTION STRING(FSP: STP) : BOOLEAN;
1033:    BEGIN STRING := FALSE;
1034:     IF FSP <> NIL THEN
1035:      WITH FSP' DO
1036:       IF FORM = ARRAYS THEN
1037:        IF COMPTYPES(AELTYPE,CHARPTR) THEN STRING := PCKDARR
1038:    END (*STRING*) ;
1039:
1040:    PROCEDURE STRINGTYPE(VAR FSP: STP);
1041:     (*ENTER TYPE OF STRINGCONST (PACKED ARRAY [1..LGTH] OF CHAR) INTO
1042:      STRUCTURE TABLE*)
1043:     VAR LSP,LSP1: STP;
1044:    BEGIN NEW(LSP,SUBRANGE);
1045:     WITH LSP' DO
1046:      BEGIN FORM := SUBRANGE; RANGETYPE := INTPTR;
1047:       MIN.IVAL := 1; MAX.IVAL := LGTH ; FTYPE := FALSE;
1048:       SIZER(LSP);
1049:      END;
1050:     NEW(LSP1,ARRAYS);
1051:     WITH LSP1' DO
1052:      BEGIN FORM := ARRAYS;
1053:       AELTYPE := CHARPTR; INXTYPE := LSP;
1054:       PCKDARR := TRUE; FTYPE := FALSE;
1055:       SIZER(LSP1)
1056:      END;
```

```
1057:     FSP := LSP1
1058:    END (*STRINGTYPE*) ;
1059:
1060:
1061:   PROCEDURE PRINTTABLES(FB: BOOLEAN);
1062:    VAR I, LIM: DISPRANGE;
1063:
1064:    PROCEDURE MARK;
1065:     VAR I: INTEGER;
1066:
1067:     PROCEDURE MARKCTP(FP: CTP); FORWARD;
1068:
1069:     PROCEDURE MARKSTP(FP: STP);
1070:     BEGIN
1071:      IF FP <> NIL THEN
1072:       WITH FP' DO
1073:        BEGIN MARKED := TRUE;
1074:         CASE FORM OF
1075:         SCALAR:   ;
1076:         SUBRANGE: MARKSTP(RANGETYPE);
1077:         POINTER:
1078:              ;
1079:         POWER:    MARKSTP(ELSET) ;
1080:         ARRAYS:   BEGIN MARKSTP(AELTYPE); MARKSTP(INXTYPE) END;
1081:         RECORDS:  BEGIN MARKCTP(FSTFLD); MARKSTP(RECVAR) END;
1082:         FILES:    MARKSTP(FILTYPE);
1083:         TAGFIELD: MARKSTP(FSTVAR);
1084:         VARIANT:  BEGIN MARKSTP(NXTVAR); MARKSTP(SUBVAR) END
1085:         END
1086:        END
1087:     END;
1088:
1089:     PROCEDURE MARKCTP;
1090:     BEGIN
1091:      IF FP <> NIL THEN
1092:       WITH FP' DO
1093:        BEGIN MARKCTP(LLINK); MARKCTP(RLINK);
1094:         MARKSTP(IDTYPE)
1095:        END
1096:     END;
1097:
1098:    BEGIN
1099:     FOR I := TOP DOWNTO LIM DO
1100:      MARKCTP(DISPLAY[I].FNAME)
1101:    END;
1102:
1103:    PROCEDURE FOLLOWCTP(FP: CTP); FORWARD;
1104:
1105:    PROCEDURE FOLLOWSTP(FP: STP);
1106:    BEGIN
1107:     IF FP <> NIL THEN
1108:      WITH FP' DO
1109:       IF MARKED THEN
1110:        BEGIN MARKED := FALSE; WRITE(OUTPUT,# #:5); WRN(ORD(FP));
1111:  (**)  (*.FOLLOWSTP. WRITE THE #SIZE# *)
1112:         CASE FORM OF
1113:         SCALAR:   BEGIN WRITE(OUTPUT,#SCALAR#:10);
1114:                IF SCALKIND = STANDARD THEN WRITE(OUTPUT,#STANDARD#:10)
1115:                ELSE
1116:                 BEGIN WRITE(OUTPUT,#DECLARED#:10,# #:4); WRCI(ORD(FCONST)) END;
1117:                WRITELN(OUTPUT)
1118:               END;
1119:         SUBRANGE: BEGIN WRITE(OUTPUT,#SUBRANGE#:10,# #:4); WRN(ORD(RANGETYPE));
1120:                IF RANGETYPE <> REALPTR THEN
1121:                 BEGIN WRCI(MIN.IVAL); WRCI(MAX.IVAL) END
1122:                ELSE
```

```
1123:                BEGIN WRCR(MIN.RVAL); WRCR(MAX.RVAL) END;
1124:               WRITELN(OUTPUT); FOLLOWSTP(RANGETYPE)
1125:              END;
1126:         POINTER:  BEGIN WRITE(OUTPUT,#POINTER#:10,# #:4); WRN(ORD(ELTYPE));
1127:              WRITELN(OUTPUT);
1128:              END;
1129:         POWER:    BEGIN IF PCKDSET THEN WRITE(OUTPUT,#PACKED#:10)
1130:                ELSE WRITE(OUTPUT,#UNPACKED#:10);
1131:              WRITE(OUTPUT,#SET#:10,# #:4); WRN(ORD(ELSET)); WRITELN(OUTPUT);
1132:  (**)        (*.POWER. PRINT ADDITIONAL INFO*)
1133:              FOLLOWSTP(ELSET);
1134:              END;
1135:         ARRAYS:   BEGIN IF PCKDARR THEN WRITE(OUTPUT,#PACKED#:10)
1136:                ELSE WRITE(OUTPUT,#UNPACKED#:10);
1137:              WRITE(OUTPUT,#ARRAY#:10,# #:4); WRN(ORD(AELTYPE)); WRSP;
1138:              WRN(ORD(INXTYPE));
1139:              WRITELN(OUTPUT);
1140:  (**)        (*.ARRAYS. PRINT ADDITIONAL INFO*)
1141:              FOLLOWSTP(AELTYPE); FOLLOWSTP(INXTYPE);
1142:              END;
1143:         RECORDS:  BEGIN IF PCKDREC THEN WRITE(OUTPUT,#PACKED#:10)
1144:                ELSE WRITE(OUTPUT,#UNPACKED#:10);
1145:              WRITE(OUTPUT,#RECORD#:10,# #:4); WRN(ORD(FIELDS)); WRSP;
1146:              WRN(ORD(FSTFLD)); WRSP; WRN(ORD(RECVAR)); WRITELN(OUTPUT);
1147:              FOLLOWCTP(FSTFLD);
1148:               FOLLOWSTP(RECVAR)
1149:              END;
1150:         FILES:    BEGIN IF PCKDFIL THEN WRITE(OUTPUT,#PACKED#:10)
1151:                ELSE WRITE(OUTPUT,#UNPACKED#:10);
1152:                WRITE(OUTPUT, #FILE#:10,# #:4); WRN(ORD(FILTYPE));
1153:                WRITELN(OUTPUT);
1154:  (**)          (*.FILES. PRINT ADDITIONAL INFO*)
1155:                FOLLOWSTP(FILTYPE);
1156:              END;
1157:         TAGFIELD: BEGIN WRITE(OUTPUT,#TAGFIELD#:10,# #:4); WRN(ORD(TGFLDP));
1158:                WRSP; WRN(ORD(FSTVAR)); WRITELN(OUTPUT);
1159:                FOLLOWSTP(FSTVAR)
1160:              END;
1161:         VARIANT:  BEGIN WRITE(OUTPUT,#VARIANT#:10,# #:4); WRN(ORD(NXTVAR));
1162:                WRSP; WRN(ORD(SUBVAR)); WRSP; WRCI(VARVAL.IVAL); WRSP;
1163:                WRCI(ORD(FSTVARFLD)); WRITELN(OUTPUT);
1164:                FOLLOWSTP(NXTVAR); FOLLOWSTP(SUBVAR)
1165:              END
1166:         END
1167:        END
1168:    END;
1169:
1170:    PROCEDURE FOLLOWCTP;
1171:     VAR I,J: INTEGER; P: CTAILP;
1172:    BEGIN
1173:     IF FP <> NIL THEN
1174:      WITH FP' DO
1175:       BEGIN WRITE(OUTPUT,# #); WRN(ORD(FP)); WRSP;
1176:        WRITE(OUTPUT,NAME:9,# #:4); WRN(ORD(LLINK)); WRSP;
1177:        WRN(ORD(RLINK)); WRSP; WRN(ORD(IDTYPE));
1178:        CASE KLASS OF
1179:         TYPES: WRITE(OUTPUT,#TYPE#:10);
1180:         KONST: BEGIN WRITE(OUTPUT,#CONSTANT#:10,# #:4); WRN(ORD(NEXT));
1181:             IF IDTYPE <> NIL THEN
1182:               IF IDTYPE = REALPTR THEN
1183:                WRCR(VALUES.RVAL)
1184:               ELSE
1185:                IF IDTYPE'.FORM = ARRAYS THEN
1186:                 BEGIN WRITE(OUTPUT,#STRING #:11);
1187:                   P := SVAL.STRP; J := 1;
1188:                   FOR I := 1 TO SVAL.STRLEN DO
```

```
1189:                  BEGIN
1190:                   IF J > STRGFRL THEN BEGIN J := 1; P := P'.NXTCSP END;
1191:                   WRITE(OUTPUT,P'.STFR[J]);
1192:                   J := J + 1;
1193:                  END;
1194:                END
1195:              ELSE WRITE(OUTPUT,VALUES.IVAL)
1196:             END;
1197:        VARS:  BEGIN WRITE(OUTPUT,#VARIABLE#:10);
1198:             IF VKIND = DRCT THEN WRITE(OUTPUT,#DRCTACC#:10)
1199:             ELSE WRITE(OUTPUT,#INDRCTACC#:10);
1200:             WRSP; WRN(ORD(NEXT)); WRCI(VLEV); WRSP; WRN(VADDR); WRSP;
1201:             WRN(PARADDR);
1202:            END;
1203:        FIELD: BEGIN WRITE(OUTPUT,#FIELD#:10,# #:4); WRN(ORD(NEXT));
1204:             IF NOT PCKDFLD THEN
1205:              BEGIN WRITE(OUTPUT,#UNPACKED#:10,# #:4); WRN(FLDADDR) END
1206:             ELSE WRPN(PFLDADDR)
1207:            END;
1208:        PROC,
1209:        FUNC:  BEGIN
1210:             IF KLASS = PROC THEN WRITE(OUTPUT,#PROCEDURE#:10)
1211:             ELSE WRITE(OUTPUT,#FUNCTION#:10);
1212:             IF PFDECKIND = STANDARD THEN
1213:              BEGIN WRITE(OUTPUT,#STANDARD#:10); WRCI(KEY) END
1214:             ELSE
1215:              BEGIN WRITE(OUTPUT,#DECLARED#:10,# #:4,ORD(NEXT):6);
1216:               WRCI(PFLEV); WRSP; WRN(PFADDR); WRSP; WRN(LCSAVE);
1217:               IF PFKIND = ACTUAL THEN
1218:                BEGIN WRITE(OUTPUT,#ACTUAL#:10);
1219:                 IF KLASS = FUNC THEN BEGIN WRSP; WRN(FCTVALADDR) END;
1220:  (**)           (*.FOLLOWCTP. PRINT ADDITIONAL INFORMATION*)
1221:                 CASE PFDECL OF
1222:                  DEF: WRITE(OUTPUT,#DEF#:10);
1223:                  NOTDEF: WRITE(OUTPUT,#NOTDEF#:10);
1224:                  FORWDECL: WRITE(OUTPUT,#FORWARD#:10);
1225:                  EXTDECL: WRITE(OUTPUT,#EXTERN#:10);
1226:                  FTNDECL: WRITE(OUTPUT,#FORTRAN#:10)
1227:                 END;
1228:  (**)           (*.FOLLOWCTP. PRINT ADDITIONAL INFORMATION FOR THE DIFFERENT
1229:                   CASES*)
1230:                END
1231:              ELSE WRITE(OUTPUT,#FORMAL#:10)
1232:              END
1233:            END
1234:         END;
1235:         WRITELN(OUTPUT); FOLLOWCTP(LLINK); FOLLOWCTP(RLINK);
1236:         FOLLOWSTP(IDTYPE)
1237:        END
1238:    END;
1239:
1240:   BEGIN
1241:    WRITELN(OUTPUT); WRITELN(OUTPUT); WRITELN(OUTPUT);
1242:    IF FB THEN LIM := 0
1243:    ELSE BEGIN LIM := TOP; WRITE(OUTPUT,# LOCAL#) END;
1244:    WRITELN(OUTPUT,# TABLES:#); WRITELN(OUTPUT);
1245:    MARK;
1246:    FOR I := TOP DOWNTO LIM DO
1247:     FOLLOWCTP(DISPLAY[I].FNAME);
1248:     WRITELN(OUTPUT);
1249:   END;
1250:
1251:   PROCEDURE NEWLOC(FSP: STP; VAR FADDR: ADDRRANGE);
1252:  (**) (*.NEWLOC. RESERVES A LOCATION FOR A VARIABLE OF TYPE #FSP#
1253:         (MEETING A POSSIBLE ALIGNEMENT CONDITION).
1254:         #FSP# MAY BE #NIL# AFTER AN ERROR.
```

```
1255:         #LC# HAS TO BE CORRECTED ACCORDINGLY.
1256:         FADDR := LC; LC := LC + FSP'.SIZE       *)  BEGIN
1257:   END (*NEWLOC*);
1258:
1259:    FUNCTION COPYCOND(FSP: STP): BOOLEAN;
1260:  (**) (*.COPYCOND. DELIVER TRUE IFF #FSP# DESCRIBES A TYPE ONE WANTS NOT TO PASS
1261:         AS A VALUE ON THE STACK INTO A CALLED PROCEDURE. IT MUST BE TRUE FOR
1262:         A TYPE WHOSE ARITHMETIC IS PERFORMED ON THE STACK*) BEGIN
1263:    END (*COPYCOND*);
1264:
1265:    PROCEDURE CONSTANT(FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU);
1266:     VAR LSP: STP; LCP: CTP; SIGN: (NONE,POS,NEG);
1267:    BEGIN LSP := NIL; FVALU.IVAL := 0;
1268:     IF NOT (SY IN CONSTBEGSYS) THEN
1269:      BEGIN ERROR(50); SKIP(FSYS+CONSTBEGSYS) END;
1270:     IF SY IN CONSTBEGSYS THEN
1271:      BEGIN
1272:       IF SY = CHARCONST THEN
1273:        BEGIN LSP := CHARPTR; FVALU.IVAL := IVAL; INSYMBOL END
1274:       ELSE
1275:        IF SY = STRINGCONST THEN
1276:         BEGIN STRINGTYPE(LSP);
1277:          FVALU.SVAL := SVAL;
1278:          INSYMBOL
1279:         END
1280:       ELSE
1281:        BEGIN
1282:         SIGN := NONE;
1283:         IF OP IN [PLUS,MINUS] THEN
1284:          BEGIN IF OP = PLUS THEN SIGN := POS ELSE SIGN := NEG;
1285:           INSYMBOL
1286:          END;
1287:         IF SY = IDENT THEN
1288:          BEGIN SEARCHID([KONST],LCP);
1289:           WITH LCP' DO
1290:            BEGIN LSP := IDTYPE; FVALU := VALUES END;
1291:           IF SIGN <> NONE THEN
1292:            IF LSP = INTPTR THEN
1293:             BEGIN IF SIGN = NEG THEN FVALU.IVAL := -FVALU.IVAL END
1294:            ELSE
1295:             IF LSP = REALPTR THEN
1296:              BEGIN
1297:               IF SIGN = NEG THEN FVALU.RVAL := -FVALU.RVAL
1298:              END
1299:            ELSE ERROR(105);
1300:           INSYMBOL;
1301:          END
1302:         ELSE
1303:          IF SY = INTCONST THEN
1304:           BEGIN IF SIGN = NEG THEN IVAL := -IVAL;
1305:            LSP := INTPTR; FVALU.IVAL := IVAL; INSYMBOL
1306:           END
1307:          ELSE
1308:           IF SY = REALCONST THEN
1309:            BEGIN IF SIGN = NEG THEN RVAL := -RVAL;
1310:             LSP := REALPTR; FVALU.RVAL := RVAL; INSYMBOL
1311:            END
1312:           ELSE
1313:            BEGIN ERROR(106); SKIP(FSYS) END
1314:        END;
1315:       IF NOT (SY IN FSYS) THEN
1316:        BEGIN ERROR(6); SKIP(FSYS) END
1317:       END;
1318:     FSP := LSP
1319:    END (*CONSTANT*) ;
1320:
```

```
1321:
1322:    PROCEDURE TYP(FSYS: SETOFSYS; VAR FSP: STP);
1323:     VAR LSP,LSP1,LSP2: STP; OLDTOP: DISPRANGE; LCP: CTP;
1324:       LMIN,LMAX: INTEGER; LSZ: SIZETYPE;
1325:       PACKFLAG,LFILTYP,EXITLOOP: BOOLEAN;
1326:       LDISPL : PADDRRANGE; (*LOCATION COUNTER WITHIN A RECORD*)
1327:       LSIZE : SIZETYPE; (*SIZE OF THE RECORD UP TO THE POINT OF PROCESSING*)
1328:
1329:     PROCEDURE INITDISPL;
1330:   (**) (*.INITDISPL. INITIALIZE #LDISPL# AND #LSIZE# TO ZERO. THE
1331:           ALIGNMENT CONDITION IS THE LEAST SEVERE ONE*) BEGIN
1332:     END (*INITDISPL*);
1333:
1334:     PROCEDURE SIMPLETYPE(FSYS: SETOFSYS; VAR FSP: STP);
1335:      VAR LSP,LSP1: STP; LCP,LCP1: CTP; TTOP: DISPRANGE;
1336:        LVAL: INTEGER; LVALU: VALU;
1337:
1338:      PROCEDURE SUBRNGS(FSP: STP; FVALU: VALU);
1339:       (*PROCESS SUBRANGE TYPE*)
1340:       VAR LOW,HIGH: INTEGER;
1341:      BEGIN NEW(LSP,SUBRANGE);
1342:       WITH LSP' DO
1343:        BEGIN RANGETYPE := FSP; FORM := SUBRANGE;
1344:         MIN := FVALU; FTYPE := FALSE
1345:        END;
1346:       IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1347:       CONSTANT(FSYS,LSP1,LVALU);
1348:       WITH LSP' DO
1349:        BEGIN MAX := LVALU;
1350:         INITSIZE(LSZ); SIZE := LSZ;
1351:         IF NOT COMPTYPES(FSP,LSP1) THEN ERROR(107)
1352:         ELSE
1353:           IF FSP = REALPTR THEN
1354:            BEGIN IF MIN.RVAL >MAX.RVAL THEN ERROR(102)
1355:            END
1356:           ELSE
1357:            IF STRING(FSP) THEN
1358:             BEGIN ERROR(148); RANGETYPE := NIL
1359:             END
1360:            ELSE
1361:             BEGIN IF MIN.IVAL>MAX.IVAL THEN ERROR(102); SIZER(LSP) END;
1362:        END
1363:      END (*SUBRNGS*);
1364:
1365:     BEGIN (*SIMPLETYPE*)
1366:      IF NOT (SY IN SIMPTYPEBEGSYS) THEN
1367:       BEGIN ERROR(1); SKIP(FSYS+SIMPTYPEBEGSYS) END;
1368:      IF SY IN SIMPTYPEBEGSYS THEN
1369:       BEGIN
1370:        IF SY = LPARENT THEN
1371:         BEGIN TTOP := TOP;   (*DECL. CONSTS LOCAL TO INNERMOST BLOCK*)
1372:          WHILE DISPLAY[TOP].OCCUR <> BLCK DO TOP := TOP - 1;
1373:          NEW(LSP,SCALAR,DECLARED);
1374:          WITH LSP' DO
1375:           BEGIN FORM := SCALAR; SCALKIND := DECLARED; FTYPE := FALSE;
1376:            FCONST := NIL; INITSIZE(LSZ); SIZE := LSZ;
1377:           END;
1378:          LCP1 := NIL; LVAL := -1;
1379:          REPEAT INSYMBOL;
1380:           IF SY = IDENT THEN
1381:            BEGIN NEW(LCP,KONST); LVAL := LVAL + 1;
1382:             WITH LCP' DO
1383:              BEGIN NAME := ID; IDTYPE := LSP; NEXT := LCP1;
1384:               VALUES.IVAL := LVAL; KLASS := KONST
1385:              END;
1386:             ENTERID(LCP);
```

```
1387:             LCP1 := LCP; INSYMBOL
1388:            END
1389:           ELSE ERROR(2);
1390:           IF NOT (SY IN FSYS+[COMMA,RPARENT]) THEN
1391:            BEGIN ERROR(6); SKIP(FSYS+[COMMA,RPARENT]) END
1392:          UNTIL SY <> COMMA;
1393:          LSP'.FCONST := LCP1;
1394:          SIZER(LSP);
1395:          TOP := TTOP;
1396:          IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
1397:         END
1398:        ELSE
1399:         BEGIN
1400:          IF SY = IDENT THEN
1401:           BEGIN SEARCHID([TYPES,KONST],LCP);
1402:            INSYMBOL;
1403:            WITH LCP' DO
1404:             IF KLASS = KONST THEN SUBRNGS(IDTYPE,VALUES)
1405:             ELSE
1406:              LSP := IDTYPE
1407:           END (*SY = IDENT*)
1408:          ELSE
1409:           BEGIN CONSTANT(FSYS+[COLON],LSP1,LVALU);
1410:            SUBRNGS(LSP1,LVALU)
1411:           END;
1412:         END;
1413:        FSP := LSP;
1414:        IF NOT (SY IN FSYS) THEN
1415:         BEGIN ERROR(6); SKIP(FSYS) END
1416:       END
1417:        ELSE FSP := NIL
1418:     END (*SIMPLETYPE*) ;
1419:
1420:     PROCEDURE FIELDLIST(FSYS: SETOFSYS; VAR FRECVAR: STP;
1421:               VAR FFSTFLD: CTP; VAR FTYP: BOOLEAN);
1422:               (* FTYP IS TRUE IFF A FIELD OF THE LIST IS OR CONTAINS A FILE *)
1423:      VAR LASTFLD, LCP,LCP1,NXT,NXT1: CTP; LSP,LSP1,LSP2,LSP3,LSP4: STP;
1424:        SAVEDISPL,MAXDISPL : PADDRRANGE; SAVESIZE,MAXSIZE: SIZETYPE;
1425:        LVALU : VALU;
1426:        TAGFLAG,LFILTYP,EXITLOOP: BOOLEAN;
1427:
1428:      PROCEDURE FIELDADDRESS(FCP: CTP; FSP: STP; FPCKDFLAG: BOOLEAN;
1429:       FLASTFLD: CTP);
1430:   (**) (*.FIELDADDRESS. COMPUTE THE DISPLACEMENT (PCKDFLD,FLDADDR,PFLDADDR) OF
1431:         THE FIELD #FCP'# OF TYPE #FSP'#.
1432:         #FSP=NIL# INDICATES THAT THE TYPE WAS NOT RECOGNIZED.
1433:         INCREASE THE GLOBAL VARIABLES #LDISPL# AND #LSIZE# BY THE REQUIRED
1434:         AMOUNT.
1435:         #FPCKDFLAG# = (THE SURROUNDING RECORD IS DECLARED AS PACKED).
1436:         #FLASTFLD# POINTS TO THE FIELD JUST BEFORE THE FIELD WE ARE PROCESSING.
1437:         IT IS #NIL# IF THE FIELD BEFORE WAS A TAGFIELD OR IF #FCP'# IS
1438:         THE FIRST FIELD IN THE RECORD.
1439:         #FLASTFLD# IS PROVIDED FOR OPTIMIZING PURPOSES: AFTER GIVING AN ADDRESS
1440:         TO THE FIELD #FCP'#, IT MAY BE USEFUL TO SIMPLIFY THE ADDRESS OF
1441:         #FLASTFLD# *) BEGIN
1442:       END (*FIELDADDRESS*);
1443:
1444:       PROCEDURE SETMAXDISPL;
1445:   (**) (*.SETMAXDISPL. MAXDISPL := MAXIMUM(LDISPL,MAXDISPL).
1446:           MAXSIZE := MAXIMUM(LSIZE,MAXSIZE).
1447:           THE ALIGNMENT CONDITION IS THE MORE SEVERE ONE *) BEGIN
1448:       END (*SETMAXDISPL*);
1449:
1450:     BEGIN (*FIELDLIST*) NXT1 := NIL; LSP := NIL;
1451:       TAGFLAG := TRUE; FTYP := FALSE; LASTFLD := NIL;
1452:       IF NOT (SY IN FSYS+[IDENT,CASESY]) THEN
```

```
1453:       BEGIN ERROR(19); SKIP(FSYS+[IDENT,CASESY]) END;
1454:      WHILE SY = IDENT DO
1455:       BEGIN NXT := NXT1;
1456:        (*LOOP UNTIL SY <> COMMA:*)
1457:        REPEAT
1458:         IF SY = IDENT THEN
1459:          BEGIN NEW(LCP,FIELD);
1460:           WITH LCP' DO
1461:            BEGIN NAME := ID; IDTYPE := NIL; NEXT := NXT;
1462:             KLASS := FIELD
1463:            END;
1464:           NXT := LCP;
1465:           ENTERID(LCP);
1466:           INSYMBOL
1467:          END
1468:         ELSE ERROR(2);
1469:         IF NOT (SY IN [COMMA,COLON]) THEN
1470:          BEGIN ERROR(6); SKIP(FSYS+[COMMA,COLON,SEMICOLON,CASESY])
1471:          END;
1472:         EXITLOOP := SY <> COMMA;
1473:         IF NOT EXITLOOP THEN INSYMBOL
1474:        UNTIL EXITLOOP;
1475:        IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1476:        TYP(FSYS+[CASESY,SEMICOLON],LSP);
1477:        WHILE NXT <> NXT1 DO
1478:         WITH NXT' DO
1479:          BEGIN IDTYPE := LSP;
1480:           IF LSP <> NIL THEN
1481:            BEGIN FTYP := FTYP OR LSP'.FTYPE;
1482:             FIELDADDRESS(NXT,LSP,PACKFLAG,LASTFLD);
1483:            END
1484:           ELSE FIELDADDRESS(NXT,NIL,PACKFLAG,LASTFLD);
1485:           TAGFLAG := FALSE; LASTFLD := NXT; NXT := NEXT
1486:          END;
1487:        NXT1 := LCP;
1488:        IF SY = SEMICOLON THEN
1489:         BEGIN INSYMBOL;
1490:          IF NOT (SY IN FSYS+[IDENT,CASESY]) THEN
1491:           BEGIN ERROR(19); SKIP(FSYS+[IDENT,CASESY]) END
1492:         END
1493:       END (*WHILE*);
1494:      NXT := NIL;
1495:      WHILE NXT1 <> NIL DO
1496:       WITH NXT1' DO
1497:        BEGIN LCP := NEXT; NEXT := NXT; NXT := NXT1; NXT1 := LCP END;
1498:      FFSTFLD := NXT;
1499:      IF SY = CASESY THEN
1500:       BEGIN NEW(LSP,TAGFIELD);
1501:        WITH LSP' DO
1502:         BEGIN TGFLDP := NIL; FSTVAR := NIL; FORM := TAGFIELD;
1503:          FTYPE := FALSE; INITSIZE(LSZ); SIZE := LSZ;
1504:         END;
1505:        FRECVAR := LSP;
1506:        INSYMBOL;
1507:        IF SY = IDENT THEN
1508:         BEGIN PRTERR := FALSE; SEARCHID([TYPES],LCP1); PRTERR := TRUE;
1509:          NEW(LCP,FIELD);
1510:          WITH LCP' DO
1511:           BEGIN IDTYPE := NIL; KLASS := FIELD; NEXT := NIL END;
1512:          IF LCP1 = NIL THEN   (*EXPLICITE TAGFIELD*)
1513:           BEGIN LCP'.NAME := ID; ENTERID(LCP);
1514:            INSYMBOL;
1515:            IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1516:            IF SY = IDENT THEN SEARCHID([TYPES],LCP1)
1517:            ELSE
1518:             BEGIN ERROR(2); SKIP(FSYS+[OFSY,LPARENT]);
```

```
1519:             LCP1 := NIL
1520:            END
1521:          END
1522:         ELSE LCP'.NAME := #        #;
1523:         IF LCP1 <> NIL THEN
1524:          BEGIN LSP1 := LCP1'.IDTYPE;
1525:           IF LSP1 <> NIL THEN
1526:            BEGIN
1527:             IF LSP1'.FORM <= SUBRANGE THEN
1528:              BEGIN
1529:               IF COMPTYPES(REALPTR,LSP1) THEN ERROR(109)
1530:               ELSE
1531:                BEGIN LSP'.TGFLDP := LCP;
1532:                 WITH LCP' DO
1533:                  BEGIN IDTYPE := LSP1;
1534:                   IF NAME <> #        # THEN
1535:                    FIELDADDRESS(LCP,LSP1,PACKFLAG,LASTFLD);
1536:                  END
1537:                END
1538:              END
1539:             ELSE ERROR(110)
1540:            END;
1541:           INSYMBOL
1542:          END
1543:        END
1544:       ELSE
1545:        BEGIN ERROR(2); SKIP(FSYS+[OFSY,LPARENT]) END;
1546:       LSP'.SIZE := LSIZE; (* SET SIZE OF THE TAGFIELD# *);
1547:       IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1548:       LSP1 := NIL; SAVEDISPL := LDISPL; MAXDISPL := LDISPL;
1549:       SAVESIZE := LSIZE; MAXSIZE := LSIZE;
1550:       (*LOOP UNTIL SY <> SEMICOLON:*)
1551:       REPEAT
1552:        IF NOT (SY IN FSYS+[SEMICOLON]) THEN
1553:        BEGIN LSP2 := NIL;
1554:        (*LOOP UNTIL SY <> SEMICOLON:*)
1555:        REPEAT CONSTANT(FSYS+[COMMA,COLON,LPARENT],LSP3,LVALU);
1556:         IF LSP'.TGFLDP <> NIL THEN
1557:          IF NOT COMPTYPES(LSP'.TGFLDP'.IDTYPE,LSP3) THEN
1558:           ERROR(111);
1559:         NEW(LSP3,VARIANT);
1560:         WITH LSP3' DO
1561:          BEGIN NXTVAR := LSP1; SUBVAR := LSP2; VARVAL := LVALU;
1562:           FORM := VARIANT; FTYPE := FALSE; INITSIZE(LSZ); SIZE := LSZ;
1563:          END;
1564:         LSP4 := LSP1;
1565:         WHILE LSP4 <> NIL DO
1566:          WITH LSP4' DO
1567:           BEGIN IF VARVAL.IVAL = LVALU.IVAL THEN ERROR(178);
1568:            LSP4 := NXTVAR;
1569:           END;
1570:         LSP1 := LSP3; LSP2 := LSP3;
1571:         EXITLOOP := SY <> COMMA;
1572:         IF NOT EXITLOOP THEN INSYMBOL
1573:        UNTIL EXITLOOP;
1574:        IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1575:        IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
1576:        FIELDLIST(FSYS+[RPARENT,SEMICOLON],LSP2,LCP,LFILTYP);
1577:        IF LFILTYP THEN ERROR(108);
1578:        FTYP := FTYP OR LFILTYP;
1579:        SETMAXDISPL;
1580:        WHILE LSP3 <> NIL DO
1581:         WITH LSP3' DO
1582:          BEGIN LSP4 := SUBVAR; SUBVAR := LSP2;
1583:           SIZE := LSIZE; (* SET THE SIZE OF THE #VARIANT# *)
1584:           FSTVARFLD := LCP;
```

```
1585:           LSP3 := LSP4
1586:          END;
1587:        IF SY = RPARENT THEN
1588:         BEGIN INSYMBOL;
1589:          IF NOT (SY IN FSYS+[SEMICOLON]) THEN
1590:           BEGIN ERROR(6); SKIP(FSYS+[SEMICOLON]) END
1591:         END
1592:        ELSE ERROR(4);
1593:        END (*NOT (SY IN ...*) ;
1594:        EXITLOOP := SY <> SEMICOLON;
1595:        IF NOT EXITLOOP THEN
1596:         BEGIN LDISPL := SAVEDISPL; LSIZE := SAVESIZE; INSYMBOL END
1597:       UNTIL EXITLOOP;
1598:       LDISPL := MAXDISPL; LSIZE := MAXSIZE;
1599:       LSP'.FSTVAR := LSP1;
1600:      END
1601:     ELSE
1602:      FRECVAR := NIL
1603:    END (*FIELDLIST*) ;
1604:
1605:   BEGIN (*TYP*) LSP := NIL;
1606:    IF NOT (SY IN TYPEBEGSYS) THEN
1607:      BEGIN ERROR(10); SKIP(FSYS+TYPEBEGSYS) END;
1608:    IF SY IN TYPEBEGSYS THEN
1609:     BEGIN
1610:      IF SY IN SIMPTYPEBEGSYS THEN SIMPLETYPE(FSYS,LSP)
1611:      ELSE
1612:   (*'*)
1613:       IF SY = ARROW THEN
1614:        BEGIN NEW(LSP,POINTER);
1615:         WITH LSP' DO
1616:          BEGIN ELTYPE := NIL; FORM := POINTER; FTYPE := FALSE;
1617:           SIZER(LSP);
1618:          END;
1619:         INSYMBOL;
1620:         IF SY = IDENT THEN
1621:          BEGIN PRTERR := FALSE;   (*NO ERROR IF SEARCH NOT SUCCESSFUL*)
1622:           SEARCHID([TYPES],LCP); PRTERR := TRUE;
1623:           IF LCP = NIL THEN   (*FORWARD REFERENCED TYPE ID*)
1624:            BEGIN NEW(LCP,TYPES);
1625:             WITH LCP' DO
1626:              BEGIN NAME := ID; IDTYPE := LSP; KLASS := TYPES;
1627:               NEXT := FWPTR
1628:              END;
1629:             FWPTR := LCP
1630:            END
1631:           ELSE
1632:            BEGIN
1633:             IF LCP'.IDTYPE <> NIL THEN
1634:              IF LCP'.IDTYPE'.FTYPE THEN ERROR(108)
1635:              ELSE LSP'.ELTYPE := LCP'.IDTYPE;
1636:            END;
1637:           INSYMBOL;
1638:          END
1639:         ELSE ERROR(2);
1640:        END
1641:       ELSE
1642:        BEGIN
1643:         IF SY = PACKEDSY THEN
1644:          BEGIN PACKFLAG := TRUE; INSYMBOL END
1645:         ELSE PACKFLAG := FALSE;
1646:         IF NOT (SY IN TYPEDELS) THEN
1647:          BEGIN ERROR(10); SKIP(FSYS+TYPEDELS) END;
1648:   (*ARRAY*)
1649:         IF SY = ARRAYSY THEN
1650:          BEGIN INSYMBOL;
```

```
1651:            IF SY = LBRACK THEN INSYMBOL ELSE ERROR(11);
1652:            LSP1 := NIL;
1653:            (*LOOP UNTIL SY <> COMMA:*)
1654:            REPEAT NEW(LSP,ARRAYS);
1655:             WITH LSP' DO
1656:              BEGIN AELTYPE := LSP1; INXTYPE := NIL;
1657:               PCKDARR := PACKFLAG; FORM := ARRAYS;
1658:               FTYPE := FALSE; INITSIZE(LSZ); SIZE := LSZ;
1659:              END;
1660:             LSP1 := LSP;
1661:             SIMPLETYPE(FSYS+[COMMA,RBRACK,OFSY],LSP2);
1662:             IF LSP2 <> NIL THEN
1663:              IF LSP2'.FORM <= SUBRANGE THEN
1664:               IF LSP2 = INTPTR THEN ERROR(149)
1665:               ELSE
1666:                IF COMPTYPES(LSP2,REALPTR) THEN ERROR(112)
1667:               ELSE LSP'.INXTYPE := LSP2
1668:              ELSE ERROR(113);
1669:             EXITLOOP := SY <> COMMA;
1670:             IF NOT EXITLOOP THEN INSYMBOL
1671:            UNTIL EXITLOOP;
1672:            IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12);
1673:            IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1674:            TYP(FSYS,LSP);
1675:            (*REVERSE POINTERS, COMPUTE SIZE *)
1676:            IF LSP <> NIL THEN
1677:             BEGIN
1678:              REPEAT
1679:               WITH LSP1' DO
1680:                BEGIN LSP2 := AELTYPE; AELTYPE := LSP;
1681:                 FTYPE := LSP'.FTYPE;
1682:                 IF INXTYPE <> NIL THEN SIZER(LSP1)
1683:                END (*WITH LSP1'*) ;
1684:               LSP := LSP1; LSP1 := LSP2
1685:              UNTIL LSP1 = NIL
1686:             END (*LSP <> NIL*)
1687:           END
1688:          ELSE
1689:    (*RECORD*)
1690:           IF SY = RECORDSY THEN
1691:            BEGIN INSYMBOL;
1692:             OLDTOP := TOP;
1693:             IF TOP < DISPLIMIT THEN
1694:              BEGIN TOP := TOP + 1;
1695:               WITH DISPLAY[TOP] DO
1696:                BEGIN FNAME := NIL; OCCUR := REC END
1697:              END
1698:             ELSE ERROR(250);
1699:             INITDISPL; (* NO OTHER CALL: COULD BE REPLACED BY INLINE
1700:                           CODE *)
1701:             FIELDLIST(FSYS-[SEMICOLON]+[ENDSY],LSP1,LCP,LFILTYP);
1702:             NEW(LSP,RECORDS);
1703:             WITH LSP' DO
1704:              BEGIN FIELDS := DISPLAY[TOP].FNAME; FTYPE := LFILTYP;
1705:               FSTFLD := LCP; RECVAR := LSP1;
1706:               PCKDREC := PACKFLAG; FORM := RECORDS
1707:              END;
1708:             LSP'.SIZE := LSIZE;
1709:             TOP := OLDTOP;
1710:             IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
1711:            END
1712:           ELSE
1713:    (*SET*)
1714:           IF SY = SETSY THEN
1715:             BEGIN INSYMBOL;
1716:              IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
```

```
1717:              SIMPLETYPE(FSYS,LSP1);
1718:              IF LSP1 <> NIL THEN
1719:               IF LSP1'.FORM > SUBRANGE THEN
1720:                BEGIN ERROR(115); LSP1 := NIL END
1721:               ELSE
1722:                IF LSP1 = REALPTR THEN ERROR(114)
1723:                ELSE
1724:                 IF LSP1 = INTPTR THEN
1725:                  ERROR(169)
1726:                 ELSE
1727:                  BEGIN GETBOUNDS(LSP1,LMIN,LMAX);
1728:                   IF (LMIN < SETMIN)OR (LMAX > SETMAX) THEN ERROR(169);
1729:                   NEW(LSP,POWER);
1730:                   WITH LSP' DO
1731:                    BEGIN FORM := POWER; FTYPE := FALSE; ELSET := LSP1;
1732:                     PCKDSET := PACKFLAG;
1733:                     SIZER(LSP)
1734:                    END
1735:                  END
1736:             END
1737:            ELSE
1738:    (*FILE*) IF SY = FILESY THEN
1739:              BEGIN INSYMBOL;
1740:               IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1741:               TYP(FSYS,LSP1);
1742:               (*COMPUTE MACHINE AND IMPLEMENTATION DEPENDANT
1743:               FILE SIZE*)
1744:               NEW(LSP,FILES);
1745:               WITH LSP' DO
1746:                BEGIN FILTYPE := LSP1; FORM := FILES; FTYPE := TRUE;
1747:                 TEXTFILE := COMPTYPES(CHARPTR,LSP1);
1748:                 IF TEXTFILE THEN PCKDFIL := TRUE
1749:                 ELSE PCKDFIL := PACKFLAG;
1750:                 (* NOTE THAT IN THIS IMPLEMENTATION ALL THE FILES OF CHAR
1751:                    ARE ASSUMED TO BE PACKED *)
1752:                 SIZER(LSP)
1753:                END;
1754:               IF LSP1 <> NIL THEN
1755:                IF LSP1'.FTYPE THEN
1756:                 BEGIN ERROR(108); LSP'.FILTYPE := NIL END;
1757:              END;
1758:          END;
1759:        IF NOT (SY IN FSYS) THEN
1760:         BEGIN ERROR(6); SKIP(FSYS) END
1761:       END;
1762:      FSP := LSP
1763:    END (*TYP*) ;
1764:
1765:    PROCEDURE LABELDECLARATION;
1766:     LABEL 1;
1767:     VAR LLP: LBP; EXITLOOP: BOOLEAN;
1768:    BEGIN
1769:     (*LOOP UNTIL SY <> COMMA:*)
1770:     REPEAT
1771:      IF SY = INTCONST THEN
1772:       BEGIN LLP := FSTLABP;
1773:        WHILE LLP <> FLABP DO
1774:         IF LLP'.LABVAL = IVAL THEN
1775:          BEGIN ERROR(166); GOTO 1 END
1776:         ELSE LLP := LLP'.NEXTLAB;
1777:        NEW(LLP);
1778:        WITH LLP' DO
1779:         BEGIN LABVAL := IVAL; DEFINED := FALSE; NEXTLAB := FSTLABP;
1780:          LABLEV := LEVEL; LCNT := 0; FSTOCC := NIL; LABCONTR := MAXINT;
1781:         END;
1782:        FSTLABP := LLP;
```

```
1783:    1:  INSYMBOL
1784:       END
1785:      ELSE ERROR(15);
1786:      IF NOT (SY IN FSYS+[COMMA,SEMICOLON]) THEN
1787:       BEGIN ERROR(6); SKIP(FSYS+[COMMA,SEMICOLON]) END;
1788:      EXITLOOP := SY <> COMMA;
1789:      IF NOT EXITLOOP THEN INSYMBOL
1790:     UNTIL EXITLOOP;
1791:     IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
1792:    END (*LABELDECLARATION*) ;
1793:
1794:    PROCEDURE CONSTDECLARATION;
1795:     VAR LCP: CTP; LSP: STP; LVALU: VALU;
1796:    BEGIN
1797:     IF SY <> IDENT THEN
1798:      BEGIN ERROR(2); SKIP(FSYS+[IDENT]) END;
1799:     WHILE SY = IDENT DO
1800:      BEGIN NEW(LCP,KONST);
1801:       WITH LCP' DO
1802:        BEGIN NAME := ID; IDTYPE := NIL; NEXT := NIL;
1803:         KLASS := KONST
1804:        END;
1805:       INSYMBOL;
1806:       IF OP = EQOP THEN INSYMBOL ELSE ERROR(16);
1807:       CONSTANT(FSYS+[SEMICOLON],LSP,LVALU);
1808:       ENTERID(LCP);
1809:       LCP'.IDTYPE := LSP; LCP'.VALUES := LVALU;
1810:       IF SY = SEMICOLON THEN
1811:        BEGIN INSYMBOL;
1812:         IF NOT (SY IN FSYS+[IDENT]) THEN
1813:          BEGIN ERROR(6); SKIP(FSYS+[IDENT]) END
1814:        END
1815:       ELSE ERROR(14)
1816:      END
1817:    END (*CONSTDECLARATION*) ;
1818:
1819:    PROCEDURE TYPEDECLARATION;
1820:     VAR LCP,LCP1,LCP2: CTP; LSP: STP; LCHCNT: INTEGER;
1821:    BEGIN
1822:     IF SY <> IDENT THEN
1823:      BEGIN ERROR(2); SKIP(FSYS+[IDENT]) END;
1824:     WHILE SY = IDENT DO
1825:      BEGIN NEW(LCP,TYPES);
1826:       WITH LCP' DO
1827:        BEGIN NAME := ID; IDTYPE := NIL; KLASS := TYPES END;
1828:       INSYMBOL;
1829:       IF OP = EQOP THEN INSYMBOL ELSE ERROR(16);
1830:       TYP(FSYS+[SEMICOLON],LSP);
1831:       ENTERID(LCP);
1832:       LCP'.IDTYPE := LSP;
1833:       (*HAS ANY FORWARD REFERENCE BEEN SATISFIED:*)
1834:       LCP1 := FWPTR;
1835:       WHILE LCP1 <> NIL DO
1836:        BEGIN
1837:         IF LCP1'.NAME = LCP'.NAME THEN
1838:          BEGIN
1839:           WITH LCP' DO
1840:            BEGIN LCP1'.IDTYPE'.ELTYPE := IDTYPE;
1841:             IF IDTYPE <> NIL THEN
1842:              IF IDTYPE'.FTYPE THEN ERROR(108)
1843:            END;
1844:           IF LCP1 <> FWPTR THEN
1845:            LCP2'.NEXT := LCP1'.NEXT
1846:           ELSE FWPTR := LCP1'.NEXT;
1847:          END;
1848:         LCP2 := LCP1; LCP1 := LCP1'.NEXT
```

```
1849:        END;
1850:       IF SY = SEMICOLON THEN
1851:        BEGIN INSYMBOL;
1852:         IF NOT (SY IN FSYS+[IDENT]) THEN
1853:          BEGIN ERROR(6); SKIP(FSYS+[IDENT]) END
1854:        END
1855:       ELSE ERROR(14)
1856:      END;
1857:     IF FWPTR <> NIL THEN
1858:      BEGIN ERROR(117); LCHCNT := CHCNT; ENDOFLINE;
1859:       REPEAT WRITELN(OUTPUT,# TYPE-ID #,FWPTR'.NAME);
1860:        FWPTR := FWPTR'.NEXT
1861:       UNTIL FWPTR = NIL;
1862:       CHCNT := LCHCNT; FOR LCHCNT := 1 TO CHCNT DO LINE[LCHCNT] := # #
1863:      END
1864:    END (*TYPEDECLARATION*) ;
1865:
1866:    PROCEDURE VARDECLARATION;
1867:     VAR LCP,NXT: CTP; LSP: STP; EXITLOOP: BOOLEAN; LCHCNT: INTEGER;
1868:      LADDR : ADDRRANGE;
1869:    BEGIN NXT := NIL;
1870:     REPEAT
1871:      (*LOOP UNTIL SY <> COMMA:*)
1872:      REPEAT
1873:       IF SY = IDENT THEN
1874:        BEGIN NEW(LCP,VARS);
1875:         WITH LCP' DO
1876:          BEGIN NAME := ID; NEXT := NXT; KLASS := VARS;
1877:           IDTYPE := NIL; VKIND := DRCT; VLEV := LEVEL
1878:          END;
1879:         ENTERID(LCP);
1880:         NXT := LCP;
1881:         INSYMBOL;
1882:        END
1883:       ELSE ERROR(2);
1884:       IF NOT (SY IN FSYS+[COMMA,COLON]+TYPEDELS) THEN
1885:        BEGIN ERROR(6); SKIP(FSYS+[COMMA,COLON,SEMICOLON]+TYPEDELS) END;
1886:       EXITLOOP := SY <> COMMA;
1887:       IF NOT EXITLOOP THEN INSYMBOL
1888:      UNTIL EXITLOOP;
1889:      IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1890:      TYP(FSYS+[SEMICOLON]+TYPEDELS,LSP);
1891:      WHILE NXT <> NIL DO
1892:       WITH  NXT' DO
1893:        BEGIN IDTYPE := LSP; NEWLOC(LSP,LADDR); VADDR := LADDR;
1894:         NXT := NEXT
1895:        END;
1896:      IF SY = SEMICOLON THEN
1897:       BEGIN INSYMBOL;
1898:        IF NOT (SY IN FSYS+[IDENT]) THEN
1899:         BEGIN ERROR(6); SKIP(FSYS+[IDENT]) END
1900:       END
1901:      ELSE ERROR(14)
1902:     UNTIL (SY <> IDENT)AND NOT (SY IN TYPEDELS);
1903:     IF FWPTR <> NIL THEN
1904:      BEGIN ERROR(117); LCHCNT := CHCNT; ENDOFLINE;
1905:       REPEAT WRITELN(OUTPUT,# TYPE-ID #,FWPTR'.NAME);
1906:        FWPTR := FWPTR'.NEXT
1907:       UNTIL FWPTR = NIL;
1908:       CHCNT := LCHCNT; FOR LCHCNT := 1 TO CHCNT DO LINE[LCHCNT] := # #
1909:      END;
1910:    END (*VARDECLARATION*) ;
1911:
1912:    PROCEDURE PROCDECLARATION(FSY: SYMBOL);
1913:     (* #FSY# WILL BE EITHER #PROCSY# OR #FUNCTSY# *)
1914:     VAR OLDLEV: 0..MAXLEVEL; LSY: SYMBOL; LCP,LCP1,PARLIST: CTP; LSP: STP;
```

```
1915:       FORW: BOOLEAN; OLDTOP: DISPRANGE;
1916:       LLC,LADDR: ADDRRANGE; LP: MARKP;
1917:  (**) (*.PROCDECLARATION. OTHER LOCAL VARIABLES*)
1918:
1919:     PROCEDURE PROCSTART(FPROCFLAG: BOOLEAN);
1920:  (**) (*.PROCSTART. THIS PROCEDURE IS CALLED AFTER DETECTING A PROCEDURE OR
1921:         FUNCTION DECLARATION.
1922:         FPROCFLAG = (IT IS A PROCEDURE) *) BEGIN
1923:     END (*PROCSTART*);
1924:
1925:     PROCEDURE PARLOC(FSP: STP; VAR FADDR: ADDRRANGE);
1926:  (**) (*.PARLOC. RESERVE A LOCATION FOR A SHORT VALUE PARAMETER OF TYPE #FSP'#
1927:         (MEETING A POSSIBLE ALIGNEMENT CONDITION).
1928:         #LC# HAS TO BE CORRECTED ACCORDINGLY.
1929:         RETURN IN #FADDR# THE VALUE OF THE ADDRESS OF THE PARAMETERS.
1930:            FADDR := LC; LC := LC + FSP'.SIZE;
1931:           *) BEGIN
1932:     END (*PARLOC*);
1933:
1934:     PROCEDURE ADDRLOC(VAR FADDR: ADDRRANGE);
1935:  (**) (*.ADDRLOC. RESERVE A LOCATION FOR AN ADDRESS (FOR LONG VALUE AND VAR
1936:         PARAMETERS). FOR OTHER COMMENTS REFER TO #PARLOC#*) BEGIN
1937:     END (*ADDRLOC*);
1938:
1939:     PROCEDURE PFLOC(VAR FADDR: ADDRRANGE);
1940:  (**) (*.PFLOC. RESERVE A LOCATION FOR A FORMAL PROC/FUNC DESCRIPTOR. FOR
1941:         OTHER COMMENTS REFER TO #PARLOC# *) BEGIN
1942:     END (*PFLOC*);
1943:
1944:     PROCEDURE PROCAFTERPARAM(FCP: CTP);
1945:  (**) (*.PROCAFTERPARAM. THIS PROCEDURE IS CALLED AFTER PROCESSING THE
1946:         PARAMETER LIST OF A FUNCTION/PROCEDURE.
1947:         #FCP# DESCRIBES THE PROCEDURE/FUNCTION *) BEGIN
1948:     END (*PROCAFTERPARAM*);
1949:
1950:     PROCEDURE FCTAFTERTYPE(FCP: CTP);
1951:  (**) (*.FCTAFTERTYPE. THIS PROCEDURE IS CALLED AFTER DETECTING THE TYPE
1952:         OF A FUNCTION*) BEGIN
1953:     END (*FCTAFTERTYPE*);
1954:
1955:     FUNCTION CORRPAR(FADDR: ADDRRANGE): ADDRRANGE;
1956:  (**) (*.CORRPAR. RETURN THE CORRECTED ADDRESS FOR A PARAMETER WHOSE PROVISIONAL
1957:         ADDRESS WAS #FADDR# *) BEGIN
1958:     END (*CORRPAR*);
1959:
1960:     FUNCTION CORRFCTVALADDR(FADDR: ADDRRANGE): ADDRRANGE;
1961:  (**) (*.CORRFCTVALADDR. RETURN THE CORRECTED VALUE FOR THE ADDRESS OF THE
1962:         EXIT VALUE OF A FUNCTION WHOSE PROVISIONAL ADDRESS WAS #FADDR#*)BEGIN
1963:     END (*CORRFCTVALADDR*);
1964:
1965:     PROCEDURE PARAMETERLIST(FSY: SETOFSYS; VAR FPAR: CTP);
1966:      VAR LCP,LCP1,LCP2,LCP3: CTP; LSP,LSP1,LSP2: STP; LKIND: DRCTINDRCT;
1967:        PACKFLAG,EXITLOOP: BOOLEAN; LADDR : ADDRRANGE;
1968:     BEGIN LCP1 := NIL;
1969:      IF NOT (SY IN FSY+[LPARENT]) THEN
1970:       BEGIN ERROR(7); SKIP(FSYS+FSY+[LPARENT]) END;
1971:      IF SY = LPARENT THEN
1972:       BEGIN IF FORW THEN ERROR(119);
1973:        INSYMBOL;
1974:        IF NOT (SY IN [IDENT,VARSY,PROCSY,FUNCTSY]) THEN
1975:         BEGIN ERROR(7); SKIP(FSYS+[IDENT,RPARENT]) END;
1976:        WHILE SY IN [IDENT,VARSY,PROCSY,FUNCTSY] DO
1977:         BEGIN
1978:          IF SY = PROCSY THEN
1979:           BEGIN
1980:            REPEAT INSYMBOL;
```

```
1981:             IF SY = IDENT THEN
1982:             BEGIN NEW(LCP,PROC,DECLARED,FORMAL);
1983:               WITH LCP' DO
1984:                BEGIN NAME := ID; IDTYPE := NIL; NEXT := LCP1;
1985:                 KLASS := PROC; PFDECKIND := DECLARED;
1986:                 PFLEV := LEVEL; PFKIND := FORMAL;
1987:                END;
1988:               ENTERID(LCP);
1989:               LCP1 := LCP;
1990:               INSYMBOL
1991:              END
1992:             ELSE ERROR(2);
1993:             IF NOT (SY IN FSYS+[COMMA,SEMICOLON,RPARENT]) THEN
1994:              BEGIN ERROR(7); SKIP(FSYS+[COMMA,SEMICOLON,RPARENT]) END
1995:            UNTIL SY <> COMMA
1996:           END
1997:          ELSE
1998:           BEGIN LCP2 := LCP1; LSP := NIL;
1999:            IF SY = FUNCTSY THEN
2000:             BEGIN
2001:              REPEAT INSYMBOL;
2002:               IF SY = IDENT THEN
2003:                BEGIN NEW(LCP,FUNC,DECLARED,FORMAL);
2004:                 WITH LCP' DO
2005:                  BEGIN NAME := ID; IDTYPE := NIL; NEXT := LCP1;
2006:                   KLASS := FUNC; PFDECKIND := DECLARED;
2007:                   PFLEV := LEVEL; PFKIND := FORMAL
2008:                  END;
2009:                 ENTERID(LCP);
2010:                 LCP1 := LCP;
2011:                 INSYMBOL
2012:                END
2013:               ELSE ERROR(2);
2014:               IF NOT (SY IN [COMMA,COLON]+FSYS) THEN
2015:                BEGIN ERROR(7); SKIP(FSYS+[COMMA,SEMICOLON,RPARENT])
2016:                END
2017:              UNTIL SY <> COMMA;
2018:              IF SY = COLON THEN
2019:               BEGIN INSYMBOL;
2020:                IF SY = IDENT THEN
2021:                 BEGIN SEARCHID([TYPES],LCP);
2022:                  LSP := LCP'.IDTYPE;
2023:                  IF LSP <> NIL THEN
2024:                   IF NOT (LSP'.FORM IN [SCALAR,SUBRANGE,POINTER])
2025:                    THEN BEGIN ERROR(120); LSP := NIL END;
2026:                  INSYMBOL
2027:                 END
2028:                ELSE ERROR(2);
2029:                IF NOT (SY IN FSYS+[SEMICOLON,RPARENT]) THEN
2030:                 BEGIN ERROR(7); SKIP(FSYS+[SEMICOLON,RPARENT]) END
2031:               END
2032:              ELSE ERROR(5)
2033:             END
2034:            ELSE
2035:             BEGIN
2036:              IF SY = VARSY THEN
2037:               BEGIN LKIND := INDRCT; INSYMBOL END
2038:              ELSE LKIND := DRCT;
2039:              (*LOOP UNTIL SY <> COMMA:*)
2040:              REPEAT
2041:               IF SY = IDENT THEN
2042:                BEGIN NEW(LCP,VARS);
2043:                 WITH LCP' DO
2044:                  BEGIN NAME := ID; IDTYPE := NIL; KLASS := VARS;
2045:                   VKIND := LKIND; NEXT := LCP1; VLEV := LEVEL;
2046:                  END;
```

```
2047:                ENTERID(LCP);
2048:                LCP1 := LCP;
2049:                INSYMBOL;
2050:               END
2051:              ELSE ERROR(2);
2052:              IF NOT (SY IN [COMMA,COLON]+FSYS) THEN
2053:               BEGIN ERROR(7); SKIP(FSYS+[COMMA,SEMICOLON,RPARENT])
2054:               END;
2055:              EXITLOOP := SY <> COMMA;
2056:              IF NOT EXITLOOP THEN INSYMBOL
2057:             UNTIL EXITLOOP;
2058:             IF SY = COLON THEN
2059:              BEGIN INSYMBOL;
2060:                IF SY = IDENT THEN
2061:                 BEGIN SEARCHID([TYPES],LCP);
2062:                  LSP := LCP'.IDTYPE;
2063:                  INSYMBOL
2064:                 END
2065:                ELSE ERROR(2);
2066:               IF LSP <> NIL THEN
2067:                IF (LKIND = DRCT) AND LSP'.FTYPE THEN
2068:                 ERROR(121);
2069:               IF NOT (SY IN FSYS+[SEMICOLON,RPARENT]) THEN
2070:                BEGIN ERROR(7); SKIP(FSYS+[SEMICOLON,RPARENT]) END
2071:              END
2072:             ELSE ERROR(5);
2073:            END;
2074:           LCP3 := LCP1;
2075:           WHILE LCP3 <> LCP2 DO
2076:            WITH LCP3' DO
2077:             BEGIN IDTYPE := LSP;
2078:              LCP3 := LCP3'.NEXT
2079:             END;
2080:          END;
2081:         IF SY = SEMICOLON THEN
2082:          BEGIN INSYMBOL;
2083:           IF NOT (SY IN FSYS+[IDENT,VARSY,PROCSY,FUNCTSY]) THEN
2084:            BEGIN ERROR(7); SKIP(FSYS+[IDENT,RPARENT]) END
2085:          END
2086:        END (*WHILE*) ;
2087:       IF SY = RPARENT THEN
2088:        BEGIN INSYMBOL;
2089:         IF NOT (SY IN FSY+FSYS) THEN
2090:          BEGIN ERROR(6); SKIP(FSY+FSYS) END
2091:        END
2092:       ELSE ERROR(4);
2093:       LCP3 := NIL;
2094:       (*REVERSE POINTERS*)
2095:       WHILE LCP1 <> NIL DO
2096:        WITH LCP1' DO
2097:         BEGIN LCP2 := NEXT; NEXT := LCP3;
2098:          LCP3 := LCP1; LCP1 := LCP2
2099:         END;
2100:       (*ADDRESS OF VAR PARAMETERS, FORMAL PROC/FUNC AND SHORT VALUE PARAMETERS*)
2101:       LCP1 := LCP3;
2102:       WHILE LCP1 <> NIL DO
2103:        WITH LCP1' DO
2104:         BEGIN
2105:          IF KLASS = VARS THEN
2106:           IF VKIND = DRCT THEN
2107:            IF COPYCOND(IDTYPE) THEN
2108:             BEGIN ADDRLOC(LADDR); LCP1'.PARADDR := LADDR END
2109:            ELSE
2110:             BEGIN PARLOC(IDTYPE,LADDR); LCP1'.VADDR := LADDR END
2111:           ELSE
2112:            BEGIN ADDRLOC(LADDR); LCP1'.VADDR := LADDR END
```

```
2113:           ELSE BEGIN PFLOC(LADDR); LCP1'.PFADDR := LADDR END;
2114:           LCP1 := NEXT;
2115:          END;
2116:        FPAR := LCP3
2117:       END
2118:        ELSE FPAR := NIL
2119:    END (*PARAMETERLIST*) ;
2120:
2121:    BEGIN (*PROCDECLARATION*)
2122:     LLC := LC;  FORW := FALSE; DP := TRUE;
2123:     PROCSTART(FSY=PROCSY);
2124:     IF SY = IDENT THEN
2125:      BEGIN SEARCHSECTION(DISPLAY[TOP].FNAME,LCP); (*DECIDE WHETHER FORW*)
2126:       IF LCP <> NIL THEN
2127:       BEGIN
2128:        LCP'.FCTVALADDR := 0;
2129:        IF LCP'.KLASS = PROC THEN
2130:         FORW := (FSY = PROCSY) AND (LCP'.PFDECL = FORWDECL)
2131:             AND (LCP'.PFKIND = ACTUAL)
2132:        ELSE
2133:         IF LCP'.KLASS = FUNC THEN
2134:          FORW := (FSY = FUNCTSY) AND (LCP'.PFDECL = FORWDECL)
2135:              AND (LCP'.PFKIND = ACTUAL)
2136:         ELSE FORW := FALSE;
2137:        IF NOT FORW THEN ERROR(160)
2138:       END
2139:       ELSE FORW := FALSE;
2140:       IF NOT FORW THEN
2141:        BEGIN
2142:         IF FSY = PROCSY THEN
2143:          BEGIN NEW(LCP,PROC,DECLARED,ACTUAL); LCP'.KLASS := PROC END
2144:         ELSE
2145:          BEGIN NEW(LCP,FUNC,DECLARED,ACTUAL); LCP'.KLASS := FUNC END;
2146:         WITH LCP' DO
2147:          BEGIN NAME := ID; IDTYPE := NIL; NEXT := NIL;
2148:           PFDECKIND := DECLARED; PFKIND := ACTUAL;
2149:           PFDECL := NOTDEF; PFLEV := LEVEL;
2150:           EXITLAB := [];
2151:           PCNT := PCNT + 1; PFCNT := PCNT; PFADDR := 0; LCSAVE := 0;
2152:  (**)     (*.PROCDECL. FILL IN ADDITIONAL INFORMATION*)
2153:          END;
2154:         ENTERID(LCP)
2155:        END;
2156:       INSYMBOL
2157:      END
2158:     ELSE
2159:      BEGIN ERROR(2); LCP := UFCTPTR END;
2160:     OLDLEV := LEVEL; OLDTOP := TOP;
2161:     IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(251);
2162:     IF TOP < DISPLIMIT THEN
2163:      BEGIN TOP := TOP + 1;
2164:       WITH DISPLAY[TOP] DO
2165:        BEGIN
2166:         IF FORW THEN FNAME := LCP'.NEXT
2167:         ELSE FNAME := NIL;
2168:         OCCUR := BLCK; PFNAME := LCP;
2169:        END
2170:      END
2171:     ELSE ERROR(250);
2172:     IF FSY = PROCSY THEN
2173:      BEGIN PARAMETERLIST([SEMICOLON],PARLIST);
2174:       IF NOT FORW THEN LCP'.NEXT := PARLIST; LSP := NIL;
2175:       PROCAFTERPARAM(LCP);
2176:      END
2177:     ELSE
2178:      BEGIN
```

```
2179:       PARAMETERLIST([SEMICOLON,COLON],PARLIST);
2180:       IF NOT FORW THEN LCP'.NEXT := PARLIST;
2181:       PROCAFTERPARAM(LCP);
2182:       IF SY = COLON THEN
2183:        BEGIN INSYMBOL;
2184:         IF SY = IDENT THEN
2185:          BEGIN IF FORW THEN ERROR(122);
2186:           SEARCHID([TYPES],LCP1);
2187:           LSP := LCP1'.IDTYPE;
2188:           LCP'.IDTYPE := LSP;
2189:           IF LSP <> NIL THEN
2190:            IF NOT (LSP'.FORM IN [SCALAR,SUBRANGE,POINTER]) THEN
2191:             BEGIN ERROR(120); LCP'.IDTYPE := NIL END;
2192:           FCTAFTERTYPE(LCP);
2193:           INSYMBOL
2194:          END
2195:         ELSE BEGIN ERROR(2); SKIP(FSYS+[SEMICOLON]) END
2196:        END
2197:       ELSE
2198:        IF NOT FORW THEN ERROR(123)
2199:      END;
2200:       (*CORRECT ADDRESS OF PARAMETERS AND FIND ADDRESS OF LONG VALUE PARAMETERS*)
2201:       LCP1 := LCP'.NEXT;
2202:       WHILE LCP1 <> NIL DO
2203:        WITH LCP1' DO
2204:         BEGIN
2205:          IF KLASS = VARS THEN
2206:           IF VKIND = DRCT THEN
2207:            BEGIN
2208:             IF IDTYPE <> NIL THEN
2209:              IF COPYCOND(IDTYPE) THEN
2210:               BEGIN NEWLOC(IDTYPE,LADDR); VADDR := LADDR;
2211:                PARADDR := CORRPAR(PARADDR);
2212:               END
2213:              ELSE VADDR := CORRPAR(VADDR);
2214:            END
2215:           ELSE VADDR := CORRPAR(VADDR)
2216:          ELSE PFADDR := CORRPAR(PFADDR);
2217:          LCP1 := NEXT;
2218:         END;
2219:       WITH LCP' DO FCTVALADDR := CORRFCTVALADDR(FCTVALADDR);
2220:     IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
2221:      IF (SY = IDENT)AND ((ID = #FORWARD #)OR (ID = #EXTERN  #)
2222:       OR (ID = #FORTRAN #)) THEN
2223:       BEGIN IF FORW THEN ERROR(161);
2224:        WITH LCP' DO
2225:         BEGIN
2226:          IF ID = #FORWARD # THEN
2227:           BEGIN PFDECL := FORWDECL; LFORWCNT := LFORWCNT + 1; LCSAVE := LC END
2228:          ELSE
2229:           IF ID = #EXTERN  # THEN PFDECL := EXTDECL
2230:           ELSE PFDECL := FTNDECL;
2231:         END;
2232:       INSYMBOL;
2233:       IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
2234:       IF NOT (SY IN FSYS) THEN
2235:        BEGIN ERROR(6); SKIP(FSYS) END
2236:      END
2237:     ELSE
2238:      BEGIN LCP'.PFDECL := NOTDEF;
2239:       IF FORW THEN BEGIN LC := LCP'.LCSAVE; LFORWCNT := LFORWCNT - 1 END;
2240:       MARK(LP);
2241:       REPEAT BLOCK(FSYS,SEMICOLON,LCP);
2242:        IF SY = SEMICOLON THEN
2243:         BEGIN INSYMBOL;
2244:          IF NOT (SY IN [BEGINSY,PROCSY,FUNCTSY]) THEN
```

```
2245:           BEGIN ERROR(6); SKIP(FSYS) END
2246:         END
2247:        ELSE ERROR(14)
2248:       UNTIL SY IN [BEGINSY,PROCSY,FUNCTSY];
2249:       RELEASE(LP);
2250:      END;
2251:     LEVEL := OLDLEV; TOP := OLDTOP; LC := LLC;
2252:    END (*PROCDECLARATION*) ;
2253:
2254:
2255:
2256:    PROCEDURE BODY(FSYS: SETOFSYS);
2257:
2258:  (**) (*.BODYCONST. DECLARATION OF CONSTANTS LOCAL TO BODY*)
2259:
2260:     TYPE
2261:
2262:      LOCFILP = ' LOCFILREC;
2263:      LOCFILREC = PACKED RECORD NXTP: LOCFILP; ADDR: PADDRRANGE;
2264:                   COUNT : INTEGER; FILETYPE: STP;
2265:                  END;
2266:
2267:
2268:
2269:
2270:                 (*TO DESCRIBE DIFFERENT THINGS*)
2271:                 (*****************************)
2272:
2273:  (**)  CNSTTYPE = (*.CNSTTYPE. DESCRIBES A CONSTANT TO BE KEPT DURING CODE
2274:                     CODE GENERATION TO AVOID UNNECESSARY REPETITIONS OF THE
2275:                     SAME CONSTANT*) INTEGER;
2276:        CNSTLIST = 'CNSTELEM;
2277:        CNSTELEM = RECORD CNST: CNSTTYPE; NEXT: CNSTLIST; OCC: LOCOFREF;
2278:                   END;
2279:        CNSTREL = (LTVAL,EQVAL,GTVAL);
2280:
2281:
2282:     VAR
2283:       LCHCNT : INTEGER; FIRST: BOOLEAN;
2284:       LCP: CTP;
2285:       LADDR : ADDRRANGE; LUSF: USFREF;
2286:       LCMAX: ADDRRANGE; (*MAX OF LC OF THE BODY*)
2287:       GLOBN: INTEGER;
2288:       FSTFORP,AVAILFORP: FORP;
2289:       EXFILP: EXTFILEP;
2290:       FSTCLSP: LOCFILP;
2291:
2292:  (**) (*.BODYDECL. DECLARATION OF VARIABLES AND PROCEDURES LOCAL TO #BODY#*)
2293:
2294:
2295:
2296:     PROCEDURE LINKOCC(VAR FPTR: LOCOFREF; FUSF: USFREF);
2297:      VAR LOCP: LOCOFREF;
2298:     BEGIN NEW(LOCP);
2299:      WITH LOCP' DO
2300:       BEGIN NXTREF := FPTR; FPTR := LOCP;
2301:        LOC := FUSF;
2302:       END
2303:     END (*LINKOCC*) ;
2304:
2305:     FUNCTION CNSTCOMP(F1CNST,F2CNST: CNSTTYPE): CNSTREL;
2306:  (**) (*.CNSTCOMP. COMPARE THE TWO KEYS #F1CNST#,#F2CNST#.
2307:         *) BEGIN
2308:     END (*CNSTCOMP*);
2309:
2310:     PROCEDURE CNSTLINK(FCNST: CNSTTYPE; FUSF: USFREF; VAR FP: CNSTLIST);
```

```
2311:     VAR LP: CNSTLIST; EXIT: BOOLEAN;
2312:      PROCEDURE ENTER(VAR FP: CNSTLIST);
2313:      VAR LP: CNSTLIST;
2314:      BEGIN NEW(LP);
2315:       WITH LP' DO
2316:        BEGIN NEXT := FP; CNST := FCNST; OCC := NIL;
2317:         LINKOCC(OCC,FUSF);
2318:        END;
2319:       FP := LP;
2320:      END (*ENTER*);
2321:     BEGIN (*CNSTLINK*)
2322:      IF FP = NIL THEN ENTER(FP)
2323:      ELSE
2324:       CASE CNSTCOMP(FP'.CNST,FCNST) OF
2325:        LTVAL: ENTER(FP);
2326:        EQVAL: LINKOCC(FP'.OCC,FUSF);
2327:        GTVAL:
2328:         BEGIN
2329:          LP := FP;
2330:          REPEAT EXIT := TRUE;
2331:           WITH LP' DO
2332:            IF NEXT = NIL THEN ENTER(NEXT)
2333:            ELSE
2334:             CASE CNSTCOMP(NEXT'.CNST,FCNST) OF
2335:              LTVAL: ENTER(NEXT);
2336:              EQVAL: LINKOCC(NEXT'.OCC,FUSF);
2337:              GTVAL: BEGIN EXIT := FALSE; LP := NEXT END;
2338:             END;
2339:          UNTIL EXIT;
2340:         END;
2341:       END;
2342:     END (*CNSTLINK*);
2343:
2344:    PROCEDURE PRINTATTR(FATTRP: ATTRP);
2345:   (**) (*.PRINTATTR. LOCAL DECLARATIONS*)
2346:    BEGIN WRITE(OUTPUT,# #);
2347:     IF FATTRP <> NIL THEN
2348:     WITH FATTRP' DO
2349:      BEGIN WRN(ORD(FATTRP)); WRSP; WRN(ORD(FOLLOW)); WRSP;
2350:       WRN(ORD(TYPTR)); WRSP; WRN(SEQNR); WRSP;
2351:   (**) (*.PRINTATTR. PRINT MACHINE DEPENDENT FIELDS*)
2352:       CASE KIND OF
2353:        CST:  BEGIN WRITE(OUTPUT,#CONSTANT#:10,# #:4);
2354:                IF TYPTR <> NIL THEN
2355:                 IF COMPTYPES(TYPTR,REALPTR) THEN WRCR(CVAL.RVAL)
2356:                 ELSE
2357:                  IF TYPTR'.FORM = POWER THEN WRITE(OUTPUT,#SET#:10)
2358:                  ELSE
2359:                   IF STRING(TYPTR) THEN WRITE(OUTPUT,#STRING#:10)
2360:                   ELSE WRCI(CVAL.IVAL);
2361:   (**)          (*.PRINTATTR. PRINT MACHINE DEPENDENT INFO*)
2362:                END;
2363:   (**)  VARBL: BEGIN (*.PRINTATTR. PRINT ADDRESS AND ACCESS INFO*)
2364:                 IF EXPR THEN
2365:                  BEGIN WRITE(OUTPUT,#EXPR#:8,# #:4);
2366:   (**)            (*.PRINTATTR. PRINT ADDRESS AND ACCESS INFO*)
2367:                  END
2368:                 ELSE
2369:                  BEGIN WRITE(OUTPUT,#NOTEXPR#:8,# #:4); WRN(ORD(FORPOINTER));
2370:                   IF INPCKDSTR THEN WRITE(OUTPUT,#PACKED#:10)
2371:                   ELSE WRITE(OUTPUT,#UNPACKED#:10);
2372:   (**)            (*.PRINTATTR. PRINT ADDRESS AND ACCESS INFO*)
2373:                  END
2374:                END
2375:       END;
2376:       WRITELN(OUTPUT)
```

```
2377:       END;
2378:     END (*PRINTATTR*);
2379:
2380:      PROCEDURE ATTRNEW;
2381:      VAR LATTRP: ATTRP;
2382:       PROCEDURE GATTRINIT;
2383:   (**) (*.GATTRINIT. INITIALIZE THE REAR END FIELDS OF #GATTRP'#*) BEGIN
2384:       END (*GATTRINIT*);
2385:      BEGIN
2386:       LATTRP := GATTRP;
2387:       IF ATTRHEAD = NIL THEN NEW(GATTRP)
2388:       ELSE
2389:        BEGIN GATTRP := ATTRHEAD; ATTRHEAD := ATTRHEAD'.FOLLOW END;
2390:       WITH GATTRP' DO
2391:        BEGIN TYPTR := NIL;
2392:         IF LATTRP = NIL THEN SEQNR := 1 ELSE SEQNR := LATTRP'.SEQNR + 1;
2393:         FOLLOW := LATTRP;
2394:         IF BATTRP = NIL THEN BATTRP := GATTRP
2395:         ELSE LATTRP'.WOLLOF := GATTRP;
2396:         WOLLOF := NIL;
2397:         GATTRINIT;
2398:        END;
2399:      END (*ATTRNEW*);
2400:
2401:      PROCEDURE RESETGATTRP(FATTRP: ATTRP);
2402:      VAR EXIT: BOOLEAN; LATTRP,LASTP: ATTRP;
2403:       PROCEDURE GATTRPDELREF;
2404:   (**) (*.GATTRPDELREF. DELETE ALL REAR END REFERENCES TO #GATTRP'#*) BEGIN
2405:       END (*GATTRPDELREF*);
2406:      BEGIN
2407:       LASTP := NIL; LATTRP := GATTRP;
2408:       EXIT := FALSE;
2409:       REPEAT
2410:        IF GATTRP = FATTRP THEN EXIT := TRUE
2411:        ELSE
2412:         IF GATTRP = NIL THEN
2413:          BEGIN EXIT := TRUE; ERROR(400) END
2414:         ELSE
2415:          BEGIN LASTP := GATTRP;
2416:           GATTRPDELREF;
2417:           GATTRP := GATTRP'.FOLLOW;
2418:          END;
2419:       UNTIL EXIT;
2420:       IF LASTP <> NIL THEN
2421:        BEGIN LASTP'.FOLLOW := ATTRHEAD; ATTRHEAD := LATTRP END;
2422:       IF GATTRP = NIL THEN BATTRP := NIL ELSE GATTRP'.WOLLOF := NIL;
2423:      END (*RESETGATTRP*);
2424:
2425:      PROCEDURE MOVATTR;
2426:      (*MOVE THE CONTENT OF #GATTRP'# TO THE ATTRIBUTE JUST BELOW IN THE ATTRIBUTE
2427:        STACK*)
2428:      VAR LATTRP,LFOLLOW,LWOLLOF: ATTRP; LSEQNR: INTEGER;
2429:       PROCEDURE REPLREFGATTRP(FATTRP: ATTRP);
2430:   (**) (*.REPLREFGATTRP. REPLACE THE REAR END REFERENCES TO #GATTRP'# BY
2431:          REFERENCES TO #FATTRP'# *) BEGIN
2432:       END (*REPLREFGATTRP*);
2433:      BEGIN
2434:       LATTRP := GATTRP'.FOLLOW;
2435:       IF LATTRP <> NIL THEN
2436:        BEGIN
2437:         REPLREFGATTRP(LATTRP);
2438:         WITH LATTRP' DO
2439:          BEGIN LFOLLOW := FOLLOW; LWOLLOF := WOLLOF; LSEQNR := SEQNR END;
2440:         LATTRP' := GATTRP';
2441:         WITH LATTRP' DO
2442:          BEGIN FOLLOW := LFOLLOW; WOLLOF := LWOLLOF; SEQNR := LSEQNR END;
```

```
2443:       END
2444:      ELSE ERROR(400);
2445:     END (*MOVATTR*);
2446:
2447:     PROCEDURE FORPUSH;
2448:     (*CREATE A NEW #FORREC# ON THE TOP OF STACK OF THE ACTIVE #FORREC# AND
2449:       FILL IT*)
2450:     VAR LFORP: FORP;
2451:     BEGIN
2452:      IF AVAILFORP = NIL THEN
2453:       NEW(LFORP)
2454:      ELSE
2455:       BEGIN
2456:        LFORP := AVAILFORP;
2457:        AVAILFORP := AVAILFORP'.NEXTFORP
2458:       END;
2459:      WITH LFORP' DO
2460:       BEGIN NEXTFORP := FSTFORP; CONTROLID := NIL
2461:       END;
2462:      FSTFORP := LFORP
2463:     END (*FORPUSH*);
2464:
2465:     PROCEDURE FORPOP;
2466:     (*RETURN TOP #FORREC#*)
2467:     VAR LFORP: FORP;
2468:     BEGIN
2469:      IF FSTFORP <> NIL THEN
2470:       BEGIN LFORP := FSTFORP;
2471:        FSTFORP := FSTFORP'.NEXTFORP;
2472:        LFORP'.NEXTFORP := AVAILFORP;
2473:        AVAILFORP := LFORP
2474:       END
2475:     END (*FORPOP*);
2476:
2477:     PROCEDURE SETLCMAX;
2478:   (**) (*.SETLCMAX. IF #LC# EXCEEDS #LCMAX# THEN SET:
2479:             LCMAX := LC *) BEGIN IF LC > LCMAX THEN LCMAX := LC;
2480:     END (*SETLCMAX*);
2481:
2482:     PROCEDURE SATISIC(FUSF: USFREF);
2483:   (**)(*.SATISIC.INSERT THE CURRENT VALUE OF THE INSTRUCTION COUNTER #IC#
2484:         IN THE PLACE DESCRIBED BY #FUSF# *) BEGIN
2485:     END (*SATISIC*);
2486:
2487:     PROCEDURE SATISALLOCC(FPTR: LOCOFREF);
2488:     BEGIN
2489:      WHILE FPTR <> NIL DO
2490:       BEGIN SATISIC(FPTR'.LOC); FPTR := FPTR'.NXTREF END;
2491:     END (*SATISALLOCC*);
2492:
2493:     PROCEDURE CHECKLABELS;
2494:     VAR LP: LBP;
2495:     BEGIN LP := FSTLABP;
2496:      WHILE LP <> FLABP DO
2497:       WITH LP' DO
2498:        BEGIN
2499:         IF DEFINED THEN
2500:          BEGIN IF GLOBN < LABCONTR THEN LABCONTR := MAXINT END
2501:         ELSE
2502:          IF FSTOCC <> NIL THEN
2503:           IF LABCONTR > GLOBN THEN LABCONTR := GLOBN;
2504:         LP := NEXTLAB;
2505:        END;
2506:     END (*CHECKLABELS*);
2507:
2508:    PROCEDURE CHECKBNDS(FMIN,FMAX: INTEGER; FERRNO:INTEGER);
```

```
2509:  (**) (*.CHECKBNDS.GENERATE TEST OF THE VALUE OF THE EXRESSION DESCRIBED BY
2510:       #GATTRP# AGAINST THE BOUNDS #FMIN# AND #FMAX#.FOR THE CASE #OUT OF BOUNDS#
2511:       GENERATE STOP OF PROGRAM EXECUTION.
2512:       ERROR KEY (FERRNO):
2513:       1 : SET ELEMENT OUT OF RANGE;
2514:       2 : VALUE OUT OF BOUNDS OF SUBRANGE TYPE;
2515:       3 : INDEX VALUE OUT OF RANGE*) BEGIN
2516:     END (*CHECKBNDS*);
2517:
2518:     PROCEDURE CHECKPOINTER(NILALLOWED: BOOLEAN);
2519:  (**) (*.CHECKPOINTER.GENERATE CODE TO CHECK THE POINTER VARIABLE DESCRIBED BY
2520:         #GATTRP#. (IT MUST HAVE A VALUE BETWEEN THE VALUE OF THE HEAP POINTER
2521:          AND THE VALUE CORRESPONDING TO THE START OF THE HEAP).
2522:         #NILALLOWED# INDICATES IF THE VALUE #NIL# IS ALLOWED*) BEGIN
2523:     END (*CHECKPOINTER*);
2524:
2525:       PROCEDURE GENJMP(FADDR:ADDRRANGE);
2526:  (**) (*.GENJMP. GENERATE A JUMP TO THE ADDRESS #FADDR#*) BEGIN
2527:       END (*GENJMP*);
2528:
2529:       PROCEDURE PREPJMP(VAR FUSF: USFREF);
2530:  (**) (*.PREPJMP. GENERATE A JUMP TO A YET UNKNOWN ADDRESS.
2531:         SAVE (IN #FUSF#) THE ADDRESS WHERE THE BRANCH ADDRESS HAS TO BE
2532:         INSERTED *) BEGIN
2533:       END (*PREPJMP*);
2534:
2535:       PROCEDURE GENFJMP(FADDR:ADDRRANGE);
2536:  (**) (*.GENFJMP. GENERATE A #FALSE JUMP# TO THE ADDRESS GIVEN BY #FADDR#
2537:        (ON #GATTRP'#) *) BEGIN
2538:       END (*GENFJMP*);
2539:
2540:       PROCEDURE PREPFJMP(VAR FUSF: USFREF);
2541:  (**) (*.PREPFJMP. GENERATE A #FALSE JUMP# TO A YET UNKNOWN ADDRESS
2542:        (ON #GATTRP'#). SAVE (IN #FUSF#) THE ADDRESS, WHERE THE BRANCH ADDRESS FOR
2543:        THE CONDITIONAL JUMP HAS TO BE INSERTED *) BEGIN
2544:       END (*PREPFJMP*);
2545:
2546:      PROCEDURE OPENFILES(FCP: CTP);
2547:       VAR EXTFILE,EXITLOOP: BOOLEAN; CLSP: LOCFILP;
2548:        LADDR: ADDRRANGE; LPADDR: PADDRRANGE;
2549:
2550:       PROCEDURE TRANSF(FADDR: ADDRRANGE; VAR FPADDR: PADDRRANGE);
2551:  (**) (*.TRANSF.GIVE #FPADDR# THE VALUE CORRESPONDING TO #FADDR#*) BEGIN
2552:       END (*TRANSF*);
2553:
2554:       PROCEDURE OPENFL(FSP: STP; FADDR: PADDRRANGE);
2555:       VAR I,LMIN,LMAX: INTEGER; LCP: CTP;
2556:
2557:         PROCEDURE RECFIELD(FSP: STP; FCP: CTP;VAR FADDR: PADDRRANGE);
2558:  (**)     (*.RECFIELD.COMPUTE THE ADDRESS OF THE FIELD #FCP# OF A RECORD OF TYPE
2559:            #FSP'# WHOSE ADDRESS IS #FADDR#. PUT THE RESULT INTO #FADDR#*) BEGIN
2560:         END (*RECFIELD*);
2561:
2562:         PROCEDURE ARRAYELEMENT(FSP: STP; FI: INTEGER; VAR FADDR: PADDRRANGE);
2563:  (**)     (*.ARRAYELEMENT.COMPUTE THE ADDRESS OF THE #FI#TH ELEMENT OF AN ARRAY
2564:            OF TYPE #FSP#. THE ADDRESS OF THE ARRAY IS INITIALLY GIVEN
2565:            BY #FADDR#. PUT THE RESULT INTO #FADDR#*) BEGIN
2566:         END (*ARRAYELEMENT*);
2567:
2568:         PROCEDURE OPENINPUT;
2569:  (**)     (*.OPENINPUT.GENERATE CODE TO OPEN THE INPUT FILE*) BEGIN
2570:         END (*OPENINPUT*);
2571:
2572:         PROCEDURE OPENOUTPUT;
2573:  (**)     (*.OPENOUTPUT.IDEM FOR THE OUTPUT FILE*) BEGIN
2574:         END (*OPENOUTPUT*);
```

```
2575:
2576:       PROCEDURE OPENEXT(FCP: CTP);
2577:  (**)   (*.OPENEXT. IDEM FOR EXTERNAL FILE DESCRIBED BY #FCP'#*) BEGIN
2578:       END (*OPENEXT*);
2579:
2580:       PROCEDURE OPENLOC(FSP: STP; FADDR: PADDRRANGE; FN: INTEGER);
2581:  (**)   (*.OPENLOC. IDEM FOR LOCAL FILE OF TYPE #FSP'#. #FN# IS THE
2582:           SPECIFIC IDENTIFICATION NUMBER OF THE FILE*) BEGIN
2583:       END (*OPENLOC*);
2584:
2585:     BEGIN (*OPENFL*)
2586:      IF FSP <> NIL THEN
2587:       WITH FSP' DO
2588:        IF FTYPE THEN
2589:        CASE FORM OF
2590:         RECORDS:
2591:          BEGIN LCP := FSTFLD;
2592:           WHILE LCP <> NIL DO
2593:            WITH LCP' DO
2594:             BEGIN
2595:              RECFIELD(FSP,LCP,FADDR);
2596:              OPENFL(IDTYPE,FADDR);
2597:              LCP:= NEXT
2598:             END
2599:          END;
2600:         ARRAYS:
2601:          IF INXTYPE <> NIL THEN
2602:           BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
2603:            FOR I := LMIN TO LMAX DO
2604:             BEGIN ARRAYELEMENT(FSP,I,FADDR); OPENFL(AELTYPE,FADDR);
2605:             END
2606:           END;
2607:         FILES:
2608:          BEGIN
2609:           IF FCP = INPUTPTR THEN OPENINPUT
2610:           ELSE
2611:            IF FCP = OUTPUTPTR THEN OPENOUTPUT
2612:            ELSE
2613:             IF EXTFILE THEN OPENEXT(FCP)
2614:             ELSE
2615:               BEGIN FILECNT := FILECNT + 1;
2616:                OPENLOC(FSP,FADDR,FILECNT);
2617:                NEW(CLSP);
2618:                WITH CLSP' DO
2619:                 BEGIN NXTP := FSTCLSP; ADDR := FADDR; COUNT := FILECNT;
2620:                  FILETYPE := FSP;
2621:                 END;
2622:                FSTCLSP := CLSP
2623:               END
2624:          END (*FILES*)
2625:        END (*CASE*)
2626:     END (*OPENFL*) ;
2627:
2628:    BEGIN (*OPENFILES*)
2629:     IF FCP <> NIL THEN
2630:      WITH FCP' DO
2631:       BEGIN OPENFILES(LLINK); OPENFILES(RLINK);
2632:        IF KLASS=VARS THEN IF VKIND=DRCT THEN
2633:         BEGIN EXTFILE := FALSE;
2634:          IF (LEVEL = 1)AND (IDTYPE <> NIL) THEN
2635:           IF IDTYPE'.FORM = FILES THEN
2636:            BEGIN EXFILP := FEXFILP; EXITLOOP := EXFILP = NIL;
2637:              WHILE NOT EXITLOOP DO
2638:              WITH EXFILP' DO
2639:               BEGIN
2640:                IF FILENAME = NAME THEN
```

```
2641:                 BEGIN EXTFILE := TRUE; DECLARED := TRUE; FILEID := FCP;
2642:                  EXITLOOP := TRUE
2643:                 END;
2644:                EXFILP := NXTP; EXITLOOP := EXITLOOP OR (EXFILP = NIL)
2645:               END
2646:             END;
2647:           LADDR := VADDR;
2648:           TRANSF(LADDR,LPADDR);
2649:           OPENFL(IDTYPE,LPADDR)
2650:          END
2651:        END (*WITH*)
2652:     END (*OPENFILES*);
2653:
2654:     PROCEDURE BODYINITIALIZE;
2655:  (**) (*.BODYINITIALIZE. LOCAL DECLARATIONS*)
2656:     BEGIN GLOBN := 0;
2657:      GATTRP := NIL; BATTRP := NIL; ATTRHEAD := NIL;
2658:      FSTFORP := NIL; AVAILFORP := NIL;
2659:      FSTCLSP := NIL;
2660:  (**)  (*.BODYINITIALIZE. MACHINE DEPENDENT INITIALIZATIONS. IN PARTICULAR:
2661:           - ALIGN #LC# TO THE VALUE OF THE STACK START
2662:           - SET THE RUNTIME STACK POINTER*)
2663:      LCMAX := LC;
2664:     END (*BODYINITIALIZE*);
2665:
2666:     PROCEDURE MAINBODYINIT;
2667:  (**) (*.MAINBODYINIT.SPECIFIC INITIALIZATIONS AT BEGINNING OF MAIN BODY.
2668:           INITIALIZE THE RUNTIME DISPLAY (IF DEFINED)*) BEGIN
2669:     END (*MAINBODYINIT*);
2670:
2671:     PROCEDURE PFBODYINIT(FPROCP: CTP);
2672:  (**) (*.PFBODYINIT.PERFORM THE WORK TO BE DONE AT THE BEGIN OF THE BODY OF
2673:           PROC/FUNC. PERFORM THE DYNAMIC LINKAGE. GIVE A VALUE TO #FPROCP.PFADDR#
2674:           AND SET #FPROCP'.PFDECL# TO #DEF#.
2675:           PERFORM THE STATIC LINKAGE IF IT WAS NOT DONE IN
2676:           #CALLNONSTANDARD#.
2677:           *) BEGIN
2678:     END (*PFBODYINIT*);
2679:
2680:     PROCEDURE OPENCODEGEN;
2681:  (**) (*.OPENCODEGEN. PREPARE CODE GENERATION OF A BODY*) BEGIN
2682:     END (*OPENCODEGEN*);
2683:
2684:     PROCEDURE CLOSECODEGEN;
2685:  (**) (*.CLOSECODEGEN. CLOSE CODE GENERATION OF A BODY*) BEGIN
2686:     END (*CLOSECODEGEN*);
2687:
2688:     PROCEDURE PREPOVFLWTEST(VAR FUSF: USFREF);
2689:  (**) (*.PREPOVFLWTEST. PREPARE THE TEST ON RUNTIME STACK OVERFLOW. RETURN
2690:           IN #FUSF# THE DESCRIPTION OF THE LOCATION WHERE #SATISOVFLW# WILL
2691:           HAVE TO MAKE AN INSERTION*) BEGIN
2692:     END (*PREPOVFLWTEST*);
2693:
2694:     PROCEDURE SATISOVFLW(FUSF: USFREF);
2695:  (**) (*.SATISOVFLW. MAKE THE INSERTION IN THE PLACE DESCRIBED BY #FUSF#.
2696:           THE GLOBAL VARIABLE #LCMAX# GIVES THE MAXIMUM VALUE OF #LC# IN THE
2697:           CURRENT PROCEDURE*) BEGIN
2698:     END (*SATISOVFLW*);
2699:
2700:     PROCEDURE LGJMPLAB(FLABP: LBP);
2701:  (**) (*.LGJMPLAB. THIS PROCEDURE IS CALLED FOR EACH LABEL JUMPED ON FROM OUTSIDE
2702:           OF THE CURRENT PROC/FUNC*) BEGIN
2703:     END (*LGJMPLAB*);
2704:
2705:     PROCEDURE COPYPARAMETERS;
2706:     VAR LCP: CTP;
```

```
2707:
2708:      PROCEDURE COPPAR(SOURCEADDR,DEST: ADDRRANGE; FSIZE: SIZETYPE);
2709:  (**)  (*.COPPAR. COPY THE VALUE FOUND IN THE LOCATION WHOSE ADDRESS IS IN
2710:           #SOURCEADDR# INTO THE LOCATION WITH ADDRESS #DEST#. THE SIZE OF THE
2711:           VALUE TO BE COPIED IS GIVEN BY #FSIZE#*) BEGIN
2712:      END (*COPPAR*);
2713:
2714:     BEGIN (*COPYPARAMETERS*)
2715:      LCP := FPROCP'.NEXT;
2716:      WHILE LCP <> NIL DO
2717:       WITH LCP' DO
2718:        BEGIN
2719:         IF VKIND = DRCT THEN
2720:          IF COPYCOND(IDTYPE) THEN COPPAR(PARADDR,VADDR,IDTYPE'.SIZE);
2721:         LCP := NEXT
2722:        END
2723:     END (*COPYPARAMETERS*);
2724:
2725:     PROCEDURE RETURN(FCP: CTP);
2726:  (**) (*.RETURN. STORE RESULT OF FUNCTION WHERE IT SHOULD BE FOR
2727:           RETURNING INTO THE CALLING PROC/FUNC (#FCP# DESCRIBES THE
2728:           CALLED FUNCTION).
2729:           GENERATE THE JUMP TO THE RETURN ADDRESS*) BEGIN
2730:     END (*RETURN*);
2731:
2732:     PROCEDURE CLOSEFILES;
2733:
2734:      PROCEDURE CLOSELOC(FSP: STP; FADDR : PADDRRANGE; FN : INTEGER);
2735:  (**) (*.CLOSELOC. GENERATE CODE TO CLOSE THE LOCAL FILE OF TYPE #FSP'# AT
2736:           ADDRESS #FADDR# WHOSE IDENTIFICATION NUMBER IS #FN# *) BEGIN
2737:      END (*CLOSELOC*);
2738:
2739:      PROCEDURE CLOSEEXT(FCP: CTP);
2740:  (**) (*.CLOSEEXT. GENERATE CODE TO CLOSE THE EXTERNAL FILE WHOSE IDENTIFIER
2741:           IS GIVEN BY #FCP'# *) BEGIN
2742:      END (*CLOSEEXT*);
2743:
2744:      PROCEDURE CLOSEINPUT;
2745:  (**) (*.CLOSEINPUT. IDEM FOR INPUT*) BEGIN
2746:      END (*CLOSEINPUT*);
2747:
2748:      PROCEDURE CLOSEOUTPUT;
2749:  (**) (*.CLOSEOUTPUT. IDEM FOR OUTPUT*) BEGIN
2750:      END (*CLOSEOUTPUT*);
2751:
2752:     BEGIN (*CLOSEFILES*);
2753:      WHILE FSTCLSP <> NIL DO
2754:       WITH FSTCLSP' DO
2755:        BEGIN CLOSELOC(FILETYPE,ADDR,COUNT); FSTCLSP := FSTCLSP'.NXTP
2756:        END;
2757:      IF LEVEL = 1 THEN
2758:       BEGIN
2759:        WHILE FEXFILP <> NIL DO
2760:         BEGIN CLOSEEXT(FEXFILP'.FILEID); FEXFILP := FEXFILP'.NXTP
2761:         END;
2762:        IF INPUTPTR <> NIL THEN CLOSEINPUT;
2763:        IF OUTPUTPTR <> NIL THEN CLOSEOUTPUT;
2764:       END
2765:     END (*CLOSEFILES*);
2766:
2767:
2768:     PROCEDURE STATEMENT(FSYS: SETOFSYS);
2769:      LABEL 1;
2770:      VAR LCP: CTP; LLP: LBP; LOCP: LOCOFREF;
2771:  (**) (*.STATEMENTDECL. OTHER LOCAL DECLARATIONS*)
2772:
```

```
2773:      PROCEDURE INTTOREAL(FATTRP: ATTRP);
2774:  (**) (*.INTTOREAL. GENERATE AN INTEGER-TO-REAL CONVERSION IN THE EXPRESSION
2775:        DESCRIBED BY #FATTRP'#.
2776:        FILL #FATTRP'# WITH THE DESCRIPTION OF THE RESULT.
2777:        IN THE CALLS OF THIS PROCEDURE, #FATTRP# POINTS EITHER TO THE TOP
2778:        ATTRIBUTE (GATTRP') OR THE ATTRIBUTE JUST BELOW THE TOP*) BEGIN
2779:       FATTRP'.TYPTR := REALPTR
2780:      END (*INTTOREAL*);
2781:
2782:      PROCEDURE EXPRESSION(FSYS: SETOFSYS); FORWARD;
2783:
2784:      PROCEDURE SELECTOR(FSYS: SETOFSYS; FCP: CTP);
2785:       VAR LATTRP: ATTRP; LCP: CTP;
2786:  (**) (*.SELECTORDECL. OTHER LOCAL DECLARATIONS FOR #SELECTOR#*)
2787:
2788:       PROCEDURE IDADDRESS;
2789:        (* PUT IN #GATTRP'# THE DESCRIPTION OF THE IDENTIFIER POINTED AT BY #FCP#.
2790:         USEFUL GLOBAL VARIABLES:
2791:          GATTRP: ATTRP (POINTS TO THE ATTRIBUTE TO BE BUILT UP)
2792:          FCP: CTP; (POINTS TO THE IDENTIFIER WE ARE WORKING ON)
2793:          DISX: DISPRANGE (IN THE CASE WHERE THE IDENTIFIER #ID# IS A FIELD
2794:                           IDENTIFIER: #DISX# IS THE LEVEL ON WHICH #ID#
2795:                           WAS DEFINED)  *)
2796:        VAR LFORP: FORP; EXITLOOP: BOOLEAN;
2797:  (**)   (*.IDADDRESS. OTHER LOCAL DECLARATIONS*)
2798:
2799:        PROCEDURE SELVARS(FCP: CTP);
2800:  (**)  (*.SELVARS. FILL THE ACCESS INFORMATION OF #GATTRP'# IN ORDER TO
2801:          ACCESS THE VARIABLE DESCRIBED BY #FCP'#:
2802:            GATTRP'.ADDRESS := FCP'. VADDR *) BEGIN
2803:        END (*SELVARS*);
2804:
2805:        PROCEDURE SELFIELD(FATTRP: ATTRP; FCP: CTP);
2806:  (**)  (*.SELFIELD. FILL THE ACCESS INFORMATION OF #GATTRP'# IN ORDER TO ACCESS
2807:          THE FIELD DESCRIBED BY #FCP'# OF A RECORD ACCESSED AS DESCRIBED BY
2808:          BY #FATTRP'#.
2809:            GATTRP'.ADDRESS := FATTRP'.ADDRESS + FCP'.FLDADDR *) BEGIN
2810:        END (*SELFIELD*);
2811:
2812:        PROCEDURE SELFCT(FCP: CTP);
2813:  (**)  (*.SELFCT. FILL THE ACCESS INFORMATION OF #GATTRP'# IN ORDER TO ACCESS
2814:          THE FUNCTION RESULT DESCRIBED BY #FCP'#.
2815:            GATTRP'.ADDRESS := FCP'.FCTVALADDR *) BEGIN
2816:        END (*SELFCT*);
2817:
2818:       BEGIN ATTRNEW;
2819:        WITH FCP', GATTRP' DO
2820:         BEGIN TYPTR := IDTYPE; KIND := VARBL; EXPR := FALSE; INPCKDSTR := FALSE;
2821:          FORPOINTER := NIL;
2822:          IF TYPTR <> NIL THEN
2823:           CASE KLASS OF
2824:            VARS:
2825:             BEGIN
2826:              SELVARS(FCP);
2827:              IF TYPTR'.FORM <= SUBRANGE THEN
2828:               IF NOT COMPTYPES(TYPTR,REALPTR) THEN
2829:                BEGIN LFORP := FSTFORP; EXITLOOP := LFORP = NIL;
2830:                 WHILE NOT EXITLOOP DO
2831:                  WITH LFORP' DO
2832:                   IF CONTROLID = FCP THEN
2833:                    BEGIN FORPOINTER := LFORP; EXITLOOP := TRUE;
2834:                    END
2835:                   ELSE
2836:                    BEGIN LFORP := LFORP'.NEXTFORP;
2837:                     EXITLOOP := LFORP = NIL
2838:                    END;
```

```
2839:                END;
2840:              END;
2841:             FIELD:
2842:              WITH DISPLAY[DISX] DO
2843:               BEGIN
2844:                SELFIELD(WITHATTRP,FCP);
2845:               END (*WITH*);
2846:             FUNC:
2847:              IF PFDECKIND = STANDARD THEN ERROR(150)
2848:              ELSE
2849:               IF PFDECL IN [EXTDECL,FTNDECL] THEN ERROR(150)
2850:               ELSE
2851:                IF PFKIND = FORMAL THEN ERROR(151)
2852:                ELSE
2853:                 IF PFLEV = LEVEL THEN ERROR(182)
2854:                 ELSE
2855:                  BEGIN
2856:                   SELFCT(FCP);
2857:                  END
2858:            END (*CASE*);
2859:          END (*WITH*);
2860:         IF P THEN
2861:          BEGIN
2862:           IF P THEN BEGIN WRITELN(OUTPUT,# IDADDRESS#); PRINTATTR(GATTRP) END;
2863:          END;
2864:       END (*IDADDRESS*) ;
2865:
2866:       PROCEDURE INDEXCODE;
2867:        (*FILL #LATTRP'# (THE ATTRIBUTE JUST BELOW THE TOP OF THE STACK) WITH
2868:          THE INFORMATION CORRESPONDING TO AN ELEMENT OF AN
2869:          ARRAY; THE ARRAY IS INITIALLY DESCRIBED BY #LATTRP'#, THE INDEX
2870:          EXPRESSION BY #GATTRP'#*)
2871:         VAR LMIN,LMAX: INTEGER;
2872:
2873:         PROCEDURE INDEXELEM(FATTRP: ATTRP);
2874:  (**)    (*.INDEXELEM. #FATTRP'# (THE ATTRIBUTE JUST BELOW THE TOP OF THE STACK)
2875:             DESCRIBES AN ARRAY.
2876:           #GATTRP'# DESCRIBES AN INDEX EXPRESSION.
2877:           REPLACE #FATTRP'# BY THE DESCRIPTION OF #ARRAY[INDEX]# *) BEGIN
2878:          WITH FATTRP' DO
2879:           BEGIN INPCKDSTR := INPCKDSTR OR TYPTR'.PCKDARR;
2880:            TYPTR := TYPTR'.AELTYPE; KIND := VARBL; EXPR := FALSE;
2881:           END;
2882:         END (*INDEXELEM*);
2883:
2884:       BEGIN
2885:         IF LATTRP'.TYPTR <> NIL THEN
2886:          GETBOUNDS(LATTRP'.TYPTR'.INXTYPE,LMIN,LMAX);
2887:         WITH GATTRP' DO
2888:          IF KIND = CST THEN
2889:           BEGIN IF (CVAL.IVAL<LMIN) OR (CVAL.IVAL>LMAX) THEN ERROR(302) END
2890:          ELSE IF DEBUG THEN CHECKBNDS(LMIN,LMAX,3);
2891:         INDEXELEM(LATTRP);
2892:         IF P THEN BEGIN WRITELN(OUTPUT,# INDEXCODE#); PRINTATTR(LATTRP) END;
2893:       END (*INDEXCODE*);
2894:
2895:       PROCEDURE RECFIELD(FCP: CTP);
2896:  (**)   (*.RECFIELD. FILL #GATTRP' WITH THE INFO CORRESPONDING TO THE RECORD
2897:           FIELD DESCRIBED BY #FCP'#; ORIGINALLY THE RECORD IS DESCRIBED
2898:           BY #GATTRP'#.
2899:               GATTRP'.ADDRESS := GATTRP'.ADDRESS + FCP'.FLDADDR*) BEGIN
2900:         WITH FCP', GATTRP' DO
2901:          BEGIN
2902:           INPCKDSTR := INPCKDSTR OR TYPTR'.PCKDREC;  TYPTR := IDTYPE;
2903:           KIND := VARBL; EXPR := FALSE;
2904:          END;
```

```
2905:        IF P THEN BEGIN WRITELN(OUTPUT,# RECFIELD#); PRINTATTR(GATTRP) END;
2906:       END (*RECFIELD*);
2907:
2908:       PROCEDURE FILEBUFFER;
2909:  (**)  (*.FILEBUFFER. FILL #GATTRP'# WITH THE INFORMATION CORRESPONDING TO
2910:           THE BUFFER ELEMENT OF THE FILE ORIGINALLY DESCRIBED BY #GATTRP'#*)BEGIN
2911:        WITH GATTRP' DO
2912:         BEGIN
2913:          INPCKDSTR := INPCKDSTR OR TYPTR'.PCKDFIL;  TYPTR := TYPTR'.FILTYPE;
2914:          KIND := VARBL; EXPR := FALSE;
2915:         END;
2916:        IF P THEN BEGIN WRITELN(OUTPUT,# FILEBUFFER#); PRINTATTR(GATTRP) END;
2917:       END (*FILEBUFFER*);
2918:
2919:       PROCEDURE POINTEDELEMENT;
2920:  (**)  (*.POINTEDELEMENT. LOCAL DECLARATIONS*)
2921:       BEGIN
2922:        WITH GATTRP' DO
2923:         BEGIN
2924:          IF DEBUG THEN CHECKPOINTER(FALSE);
2925:  (**)    (*.POINTEDELEMENT. FILL #GATTRP'# WITH THE INFO CORRESPONDING TO THE
2926:            ELEMENT POINTED AT BY THE POINTER VARIABLE ORIGINALLY DESCRIBED BY
2927:            #GATTRP'#*)
2928:          INPCKDSTR := FALSE;
2929:          TYPTR := TYPTR'.ELTYPE; KIND := VARBL; EXPR := FALSE;
2930:         END;
2931:        IF P THEN BEGIN WRITELN(OUTPUT,# POINTEDELEMENT#); PRINTATTR(GATTRP) END;
2932:       END (*POINTEDELEMENT*);
2933:
2934:      BEGIN (*SELECTOR*)
2935:       IDADDRESS;
2936:       IF NOT (SY IN SELECTSYS+FSYS) THEN
2937:        BEGIN ERROR(59); SKIP(SELECTSYS+FSYS) END;
2938:       WHILE SY IN SELECTSYS DO
2939:        BEGIN
2940:  (*[*)   IF SY = LBRACK THEN
2941:          BEGIN
2942:           REPEAT
2943:            WITH GATTRP' DO
2944:             IF TYPTR <> NIL THEN
2945:              IF TYPTR'.FORM <> ARRAYS THEN
2946:               BEGIN ERROR(138); TYPTR := NIL END;
2947:            LATTRP := GATTRP;
2948:            INSYMBOL; EXPRESSION(FSYS+[COMMA,RBRACK]);
2949:            IF GATTRP'.TYPTR <> NIL THEN
2950:             IF GATTRP'.TYPTR'.FORM > SUBRANGE THEN ERROR(113);
2951:            IF LATTRP'.TYPTR <> NIL THEN
2952:             WITH LATTRP'.TYPTR' DO
2953:              BEGIN
2954:               IF COMPTYPES(INXTYPE,GATTRP'.TYPTR) THEN
2955:                BEGIN
2956:                 IF (INXTYPE <> NIL)AND (AELTYPE <> NIL) THEN INDEXCODE
2957:                END
2958:               ELSE ERROR(139);
2959:               RESETGATTRP(LATTRP);
2960:              END
2961:           UNTIL SY <> COMMA;
2962:           IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
2963:          END (*IF SY = LBRACK*)
2964:         ELSE
2965:  (*.*)    IF SY = PERIOD THEN
2966:           BEGIN
2967:            WITH GATTRP' DO
2968:             BEGIN
2969:              IF TYPTR <> NIL THEN
2970:               IF TYPTR'.FORM <> RECORDS THEN
```

```
2971:               BEGIN ERROR(140); TYPTR := NIL END;
2972:             INSYMBOL;
2973:             IF SY = IDENT THEN
2974:              BEGIN
2975:               IF TYPTR <> NIL THEN
2976:                BEGIN SEARCHSECTION(TYPTR'.FIELDS,LCP);
2977:                 IF LCP = NIL THEN
2978:                  BEGIN ERROR(152); TYPTR := NIL END
2979:                 ELSE
2980:                  RECFIELD(LCP);
2981:                END;
2982:               INSYMBOL
2983:              END (*SY = IDENT*)
2984:             ELSE ERROR(2)
2985:            END (*WITH GATTRP'*)
2986:          END (*IF SY = PERIOD*)
2987:         ELSE
2988:  (*'*)   BEGIN
2989:           IF GATTRP'.TYPTR <> NIL THEN
2990:            BEGIN
2991:             WITH GATTRP'.TYPTR' DO
2992:              IF FORM = FILES THEN FILEBUFFER
2993:              ELSE
2994:               IF FORM = POINTER THEN POINTEDELEMENT
2995:               ELSE ERROR(141);
2996:            END;
2997:           INSYMBOL
2998:          END;
2999:        IF NOT (SY IN FSYS+SELECTSYS) THEN
3000:         BEGIN ERROR(6); SKIP(FSYS+SELECTSYS) END
3001:       END (*WHILE*) ;
3002:      IF P THEN BEGIN WRITELN(OUTPUT,# SELECTOR#); PRINTATTR(GATTRP) END;
3003:     END (*SELECTOR*) ;
3004:
3005:     PROCEDURE CALL(FSYS: SETOFSYS; FCP: CTP);
3006:      VAR LKEY: 1..NRSTDNAMES;
3007:
3008:      PROCEDURE VARIABLE(FSYS: SETOFSYS);
3009:       VAR LCP: CTP;
3010:      BEGIN
3011:       IF SY = IDENT THEN
3012:        BEGIN SEARCHID([VARS,FIELD],LCP); INSYMBOL END
3013:       ELSE BEGIN ERROR(2); LCP := UVARPTR END;
3014:       SELECTOR(FSYS,LCP)
3015:      END (*VARIABLE*) ;
3016:
3017:      PROCEDURE STANDFILE(FCP: CTP);
3018:      (*SET INTO A NEW #GATTRP'# THE DESCRIPTION OF THE FILE GIVEN BY
3019:        #FCP'#. #FCP# MAY BE EQUAL TO #INPUTPTR#, #OUTPUTPTR# OR NIL*)
3020:  (**)  (*.STANDFILE. LOCAL DECLARATIONS*)
3021:      BEGIN
3022:       ATTRNEW;
3023:       WITH GATTRP' DO
3024:        BEGIN TYPTR := TEXTPTR; KIND := VARBL; EXPR := FALSE; INPCKDSTR := FALSE;
3025:         FORPOINTER := NIL;
3026:         IF FCP = NIL THEN ERROR(175)
3027:         ELSE
3028:          IF FCP = INPUTPTR THEN
3029:  (**)     (*.STANDFILE. FILL ADDRESS AND ACCESS INFO CORRESPONDING TO FILE
3030:             #INPUT# *)
3031:          ELSE
3032:           IF FCP = OUTPUTPTR THEN
3033:  (**)      (*.STANDFILE. FILL ADDRESS AND ACCESS INFO CORRESPONDING TO FILE
3034:              #OUTPUT# *)
3035:           ELSE ERROR(400)
3036:        END
```

```
3037:       END (*SETSTFILATTR*);
3038:
3039:       PROCEDURE STDFLPROCS;
3040:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3041:        VARIABLE(FSYS+[COMMA,RPARENT]);
3042:        IF GATTRP'.TYPTR <> NIL THEN
3043:         IF GATTRP'.TYPTR'.FORM = FILES THEN
3044:  (**)    (*.STDFLPROCS. GENERATE CODE TO PERFORM THE STANDARD FILE PROCEDURE ON
3045:             THE FILE DESCRIBED BY #GATTRP'#.
3046:             KEY OF THE PROCEDURE IN #LKEY#:
3047:             1:GET, 2:PUT, 3:RESET, 4:REWRITE *)
3048:           ELSE ERROR(116);
3049:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3050:       END (*STDFLPROCS*) ;
3051:
3052:       PROCEDURE READ;
3053:        VAR FILATTRP,LATTRP: ATTRP;
3054:          GETIN,EXITLOOP: BOOLEAN;
3055:
3056:        PROCEDURE READLN;
3057:  (**)  (*.READLN. GENERATE A READ-TO-EOLN OPERATION ON THE FILE DESCRIBED BY
3058:          #FILATTRP'# *) BEGIN
3059:        END (*READLN*);
3060:
3061:        PROCEDURE READCHAR;
3062:  (**)  (*.READCHAR. GENERATE CODE FOR A READ OPERATION FROM A FILE DESCRIBED BY
3063:          #FILATTRP'# INTO A VARIABLE OF THE TYPE CHAR DESCRIBED BY #GATTRP '#
3064:         *) BEGIN
3065:        END (*READCHAR*);
3066:
3067:        PROCEDURE READINT;
3068:  (**)   (*.READINT. IDEM FOR A VARIABLE OF TYPE INT*) BEGIN
3069:       END (*READINT*);
3070:
3071:        PROCEDURE READREAL;
3072:  (**)  (*.READREAL. IDEM FOR A VARIABLE OF TYPE REAL*) BEGIN
3073:       END (*READREAL*);
3074:
3075:      BEGIN (*READ*)
3076:       (*SET DEFAULT FILE ATTRIBUTES:*)
3077:       LATTRP := GATTRP;
3078:       STANDFILE(INPUTPTR); FILATTRP := GATTRP;
3079:       IF SY = LPARENT THEN
3080:        BEGIN GETIN := TRUE;
3081:         INSYMBOL; VARIABLE(FSYS+[COMMA,RPARENT]);
3082:         IF GATTRP'.TYPTR <> NIL THEN
3083:          IF GATTRP'.TYPTR'.FORM = FILES THEN
3084:           BEGIN
3085:            IF NOT GATTRP'.TYPTR'.TEXTFILE AND (LKEY = 7 (*READLN*)) THEN
3086:             ERROR(116);
3087:            MOVATTR; RESETGATTRP(FILATTRP);
3088:            IF SY = RPARENT THEN
3089:            BEGIN IF LKEY = 6 (*READ*) THEN ERROR(116);
3090:             GETIN := FALSE
3091:            END
3092:            ELSE
3093:            IF SY = COMMA THEN
3094:             BEGIN INSYMBOL; VARIABLE(FSYS+[COMMA,RPARENT]) END
3095:           END (*FORM = FILES*);
3096:         IF GETIN THEN
3097:          (*LOOP UNTIL SY <> COMMA:*)
3098:          REPEAT
3099:           IF COMPTYPES(GATTRP'.TYPTR,CHARPTR) THEN
3100:            BEGIN IF GATTRP'.FORPOINTER <> NIL THEN ERROR(179); READCHAR END
3101:           ELSE
3102:            IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN
```

```
3103:              BEGIN
3104:               IF GATTRP'.FORPOINTER <> NIL THEN ERROR(179);
3105:               READINT
3106:              END
3107:             ELSE
3108:              IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN READREAL
3109:              ELSE ERROR(116);
3110:             RESETGATTRP(FILATTRP);
3111:            EXITLOOP := SY <> COMMA;
3112:            IF NOT EXITLOOP THEN
3113:             BEGIN INSYMBOL; VARIABLE(FSYS+[COMMA,RPARENT]) END
3114:           UNTIL EXITLOOP;
3115:          IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3116:         END (*SY = LPARENT*)
3117:        ELSE
3118:         IF LKEY = 6 THEN (*READ*) ERROR(116);
3119:        IF LKEY = 7 (*READLN*) THEN READLN;
3120:       RESETGATTRP(LATTRP);
3121:       END (*READ*) ;
3122:
3123:       PROCEDURE WRITE;
3124:        VAR FORM1P,FORM2P,FILATTRP,SAVATTRP,VALATTRP,LATTRP: ATTRP;
3125:         PUTOUT,EXITLOOP: BOOLEAN; LSP: STP;
3126:
3127:        PROCEDURE WRITELN;
3128:  (**)   (*.WRITELN. GENERATE A WRITE-EOL OPERATION ON THE FILE DESCRIBED BY
3129:             #FILATTRP'# *) BEGIN
3130:        END (*WRITELN*);
3131:
3132:        PROCEDURE WRITECHAR;
3133:  (**)  (*.WRITECHAR. GENERATE A WRITE OPERATION ON THE FILE DESCRIBED BY
3134:          #FILATTRP'#. THE EXPRESSION TO BE OUTPUT IS OF TYPE CHAR AND IS
3135:          DESCRIBED BY #VALATTR#. #FORM1P'# DESCRIBES THE LENGTH OF THE FIELD.
3136:          THE ATTRIBUTE WERE PUT ONTO THE STACK IN THE FOLLOWING ORDER:
3137:           FILATTRP, VALATTRP, FORM1P.
3138:          FORM1P = NIL IMPLIES STANDARD LENGTH *) BEGIN
3139:        END (*WRITECHAR*);
3140:
3141:        PROCEDURE WRITEINT;
3142:  (**)  (*.WRITEINT. IDEM FOR AN INTEGER*) BEGIN
3143:        END (*WRITEINT*);
3144:
3145:        PROCEDURE WRITEBOOL;
3146:  (**)  (*.WRITEBOOL. IDEM FOR A BOOLEAN*) BEGIN
3147:        END (*WRITEBOOL*);
3148:
3149:        PROCEDURE WRITESTRING;
3150:  (**) (*.WRITESTRING. IDEM FOR A STRING*) BEGIN
3151:        END (*WRITESTRING*);
3152:
3153:        PROCEDURE WRITEREAL;
3154:  (**)  (*.WRITEREAL. IDEM FOR A REAL. THE ATTRIBUTE #FORM2P'# DESCRIBES THE
3155:          NUMBER OF DIGITS AFTER THE DECIMAL POINT.
3156:          #FORM2P'# IS ON THE TOP OF THE STACK. #FORM2P = NIL# IMPLIES
3157:          #E# - FORMAT*) BEGIN
3158:        END (*WRITEREAL*);
3159:
3160:       BEGIN (*WRITE*)
3161:        (*SET DEFAULT FILE ATTRIBUTES:*)
3162:        LATTRP := GATTRP; STANDFILE(OUTPUTPTR); FILATTRP := GATTRP;
3163:        IF SY = LPARENT THEN
3164:         BEGIN PUTOUT := TRUE;
3165:          INSYMBOL; EXPRESSION(FSYS+[COMMA,COLON,RPARENT,IDENT]);
3166:          IF GATTRP'.TYPTR <> NIL THEN
3167:           IF GATTRP'.TYPTR'.FORM = FILES THEN
3168:            BEGIN
```

```
3169:             IF NOT GATTRP'.TYPTR'.TEXTFILE AND (LKEY=9 (*WRITELN*)) THEN
3170:              ERROR(116);
3171:             IF SY = RPARENT THEN
3172:              BEGIN IF LKEY = 8 (*WRITE*) THEN ERROR(116);
3173:               PUTOUT := FALSE
3174:              END
3175:             ELSE
3176:              IF SY <> COMMA THEN
3177:               BEGIN ERROR(116); SKIP(FSYS+[COMMA,RPARENT]) END;
3178:             MOVATTR; RESETGATTRP(FILATTRP);
3179:             IF SY = COMMA THEN
3180:              BEGIN INSYMBOL;
3181:               EXPRESSION(FSYS+[COMMA,COLON,RPARENT,IDENT])
3182:              END
3183:            END (*FORM = FILES*) ;
3184:          IF PUTOUT THEN
3185:           (*LOOP UNTIL SY <> COMMA*)
3186:           REPEAT
3187:            LSP := GATTRP'.TYPTR;
3188:            VALATTRP := GATTRP; FORM1P:=NIL; FORM2P := NIL;
3189:            IF SY = COLON THEN
3190:             BEGIN INSYMBOL; EXPRESSION(FSYS+[COMMA,COLON,RPARENT,IDENT]);
3191:              IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN FORM1P := GATTRP
3192:              ELSE ERROR(116);
3193:              IF SY = COLON THEN
3194:               BEGIN INSYMBOL;
3195:                EXPRESSION(FSYS+[COMMA,RPARENT]);
3196:                IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN
3197:                 IF COMPTYPES(LSP,REALPTR) THEN FORM2P := GATTRP
3198:                 ELSE ERROR(124)
3199:                ELSE ERROR(116);
3200:               END
3201:             END;
3202:             IF COMPTYPES(LSP,CHARPTR)THEN WRITECHAR
3203:             ELSE
3204:              IF COMPTYPES(LSP,INTPTR) THEN WRITEINT
3205:              ELSE
3206:               IF COMPTYPES(LSP,REALPTR) THEN WRITEREAL
3207:               ELSE
3208:                IF COMPTYPES(LSP,BOOLPTR) THEN WRITEBOOL
3209:                ELSE
3210:                 IF LSP <> NIL THEN
3211:                  IF STRING(LSP) THEN WRITESTRING
3212:                  ELSE ERROR(116);
3213:            RESETGATTRP(FILATTRP);
3214:            EXITLOOP := SY <> COMMA;
3215:            IF NOT EXITLOOP THEN
3216:             BEGIN INSYMBOL;
3217:              EXPRESSION(FSYS+[COMMA,COLON,RPARENT,IDENT])
3218:             END;
3219:           UNTIL EXITLOOP;
3220:          IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3221:         END (*SY = LPARENT*)
3222:        ELSE
3223:         IF LKEY = 8 THEN (*WRITE*) ERROR(116);
3224:        IF LKEY = 9 THEN WRITELN;
3225:        RESETGATTRP(LATTRP);
3226:       END (*WRITE*) ;
3227:
3228:       PROCEDURE PAGE;
3229:       VAR LATTRP: ATTRP;
3230:
3231:        PROCEDURE PAGEGEN;
3232:  (**)  (*.PAGEGEN. GENERATE CODE FOR PRODUCING A NEW PAGE ON THE FILE DES-
3233:          RIBED BY #GATTRP'# *) BEGIN
3234:        END (*PAGEGEN*);
```

```
3235:
3236:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3237:        LATTRP := GATTRP;
3238:        VARIABLE(FSYS+[RPARENT]);
3239:        IF GATTRP'.TYPTR <> NIL THEN
3240:         BEGIN
3241:          WITH GATTRP'.TYPTR' DO
3242:           IF FORM = FILES THEN
3243:            BEGIN IF NOT TEXTFILE THEN ERROR(116);
3244:             PAGEGEN;
3245:            END
3246:           ELSE ERROR(116)
3247:         END;
3248:        RESETGATTRP(LATTRP);
3249:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3250:       END (*PAGE*) ;
3251:
3252:      PROCEDURE PACK;
3253:       VAR SOURCEP,INDEXP,DESTP,LATTRP: ATTRP;
3254:         LSP,LSP1: STP; LOW,HIGH,LMIN,LMAX: INTEGER;
3255:
3256:         PROCEDURE PACKCHECK(FATTRP: ATTRP; MIN,MAX: INTEGER);
3257:  (**)   (*.PACKCHECK. GENERATE CODE TO CHECK #FATTRP'# AGAINST THE BOUNDS
3258:           GIVEN BY #MIN# AND #MAX#. # FATTRP'# IS JUST BELOW THE TOP OF THE
3259:           STACK *) BEGIN
3260:         END (*PACKCHECK*);
3261:
3262:         PROCEDURE PACKGEN(FSOURCEP,FINDEXP,FDESTP: ATTRP);
3263:  (**)   (*.PACKGEN. GENERATE CODE FOR THE PACK OPERATION WITH PARAMETERS
3264:           #FSOURCEP'#, #FINDEXP'#, #FDESTP'#. THE PARAMETERS ARE
3265:           IN THE THIRD, SECOND, FIRST ELEMENT FROM THE TOP OF THE STACK*) BEGIN
3266:         END (*PACKGEN*);
3267:
3268:      BEGIN (*PACK*)
3269:       LATTRP := GATTRP;
3270:       IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3271:       VARIABLE(FSYS+[COMMA,RPARENT]); SOURCEP := GATTRP;
3272:       LSP := NIL; LSP1 := NIL; LOW := 0; HIGH := 0;
3273:       IF SOURCEP'.TYPTR <> NIL THEN
3274:        WITH SOURCEP'.TYPTR' DO
3275:         IF FORM = ARRAYS THEN
3276:          IF  NOT PCKDARR THEN
3277:           BEGIN LSP := INXTYPE; LSP1 := AELTYPE;
3278:            IF LSP <> NIL THEN GETBOUNDS(LSP,LOW,HIGH);
3279:           END
3280:          ELSE ERROR(116)
3281:         ELSE ERROR(116);
3282:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3283:       EXPRESSION(FSYS+[COMMA,RPARENT]);
3284:       INDEXP := GATTRP;
3285:       IF NOT COMPTYPES(INDEXP'.TYPTR,LSP) THEN ERROR(116);
3286:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3287:       VARIABLE(FSYS+[RPARENT]);
3288:       DESTP := GATTRP;
3289:       IF GATTRP'.TYPTR <> NIL THEN
3290:        WITH GATTRP'.TYPTR' DO
3291:         IF FORM = ARRAYS THEN
3292:          IF PCKDARR AND
3293:           COMPTYPES(AELTYPE,LSP1) AND COMPTYPES(INXTYPE,LSP)
3294:          THEN
3295:           BEGIN LMIN := 0; LMAX := 0;
3296:            IF INXTYPE <> NIL THEN GETBOUNDS(INXTYPE,LMIN,LMAX);
3297:            IF LMAX - LMIN > HIGH - LOW THEN ERROR(116)
3298:            ELSE
3299:             WITH INDEXP' DO
3300:              IF KIND = CST THEN
```

```
3301:               BEGIN
3302:                IF (CVAL.IVAL<LOW) OR (CVAL.IVAL>HIGH-LMAX+LMIN) THEN ERROR(116)
3303:               END
3304:              ELSE
3305:               IF DEBUG THEN PACKCHECK(INDEXP,LOW,HIGH-LMAX+LMIN);
3306:            END
3307:           ELSE ERROR(116)
3308:          ELSE ERROR(116);
3309:       PACKGEN(SOURCEP,INDEXP,DESTP);
3310:       RESETGATTRP(LATTRP);
3311:       IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3312:      END (*PACK*) ;
3313:
3314:      PROCEDURE UNPACK;
3315:       VAR
3316:         LSP,LSP1: STP; SOURCEP,DESTP,INDEXP,LATTRP: ATTRP;
3317:         LOW,HIGH,LMIN,LMAX: INTEGER;
3318:
3319:       PROCEDURE UNPACKGEN(SOURCEP,DESTP,INDEXP: ATTRP);
3320:  (**) (*.UNPACKGEN. GENERATE AN #UNPACK# OPERATION ON THE ARGUMENTS
3321:        #SOURCEP'#, #DESTP'#, #INDEXP'# (ON THE TOP OF THE STACK IN THIS
3322:        ORDER) *) BEGIN
3323:       END (*UNPACKGEN*);
3324:
3325:      BEGIN (*UNPACK*)
3326:       LATTRP := GATTRP;
3327:       IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3328:       EXPRESSION(FSYS+[COMMA,RPARENT]);
3329:       SOURCEP := GATTRP;
3330:       LSP := NIL; LSP1 := NIL; LMIN := 0; LMAX := 0;
3331:       IF GATTRP'.TYPTR <> NIL THEN
3332:        WITH GATTRP'.TYPTR' DO
3333:         IF FORM = ARRAYS THEN
3334:          IF PCKDARR THEN
3335:           BEGIN LSP := INXTYPE; LSP1 := AELTYPE;
3336:            IF LSP <> NIL THEN GETBOUNDS(LSP,LMIN,LMAX);
3337:           END
3338:          ELSE ERROR(116)
3339:         ELSE ERROR(116);
3340:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3341:       VARIABLE(FSYS+[COMMA,RPARENT]);
3342:       DESTP := GATTRP;
3343:       IF DESTP'.TYPTR <> NIL THEN
3344:        WITH DESTP'.TYPTR' DO
3345:         IF FORM = ARRAYS THEN
3346:          IF NOT PCKDARR AND COMPTYPES(AELTYPE,LSP1)
3347:           AND COMPTYPES(INXTYPE,LSP) THEN
3348:           BEGIN LOW := 0; HIGH := 0;
3349:            IF INXTYPE <> NIL THEN GETBOUNDS(INXTYPE,LOW,HIGH);
3350:            IF LMAX - LMIN > HIGH - LOW THEN ERROR(116);
3351:           END
3352:          ELSE ERROR(116)
3353:         ELSE ERROR(116);
3354:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
3355:       EXPRESSION(FSYS+[RPARENT]);
3356:       INDEXP := GATTRP;
3357:       IF NOT COMPTYPES(GATTRP'.TYPTR,LSP) THEN ERROR(116)
3358:       ELSE
3359:        WITH GATTRP' DO
3360:         IF KIND = CST THEN
3361:          BEGIN
3362:           IF (CVAL.IVAL<LOW) OR (CVAL.IVAL>HIGH-LMAX+LMIN) THEN ERROR(116)
3363:          END
3364:         ELSE IF DEBUG THEN CHECKBNDS(LOW,HIGH-LMAX+LMIN,2);
3365:       UNPACKGEN(SOURCEP,DESTP,INDEXP);
3366:       RESETGATTRP(LATTRP);
```

```
3367:       IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3368:      END (*UNPACK*);
3369:
3370:       PROCEDURE NEWPROC;
3371:        LABEL 1;
3372:        VAR LSP,LSP1: STP; LVAL: VALU; LSIZE: SIZETYPE; LMIN,LMAX: INTEGER;
3373:         LATTRP : ATTRP;
3374:        PROCEDURE NEWGEN(FSIZE: SIZETYPE);
3375:  (**)  (*.NEWGEN. GENEARATE A #NEW# OPERATION AND STORE THE POINTER VALUE AT THE
3376:          PLACE DESCRIBED BY #GATTRP'#. THE SIZE OF THE ELEMENT TO BE CREATED
3377:          IS #FSIZE# *) BEGIN
3378:        END (*NEWGEN*);
3379:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3380:        LATTRP := GATTRP;
3381:        VARIABLE(FSYS+[COMMA,RPARENT]);
3382:        LSP := NIL; INITSIZE(LSIZE);
3383:        IF GATTRP'.TYPTR <> NIL THEN
3384:         WITH GATTRP'.TYPTR' DO
3385:          IF FORM = POINTER THEN
3386:           BEGIN
3387:            IF ELTYPE <> NIL THEN
3388:             BEGIN LSIZE := ELTYPE'.SIZE;
3389:              IF ELTYPE'.FORM = RECORDS THEN LSP := ELTYPE'.RECVAR
3390:             END
3391:           END
3392:          ELSE ERROR(116);
3393:        WHILE SY = COMMA DO
3394:         BEGIN INSYMBOL; CONSTANT(FSYS+[COMMA,RPARENT],LSP1,LVAL);
3395:          IF LSP = NIL THEN ERROR(158)
3396:          ELSE
3397:            IF LSP'.TGFLDP <> NIL THEN
3398:             IF (LSP1 = REALPTR) OR STRING(LSP1) THEN ERROR(159)
3399:             ELSE
3400:              IF COMPTYPES(LSP'.TGFLDP'.IDTYPE,LSP1) THEN
3401:               BEGIN
3402:                GETBOUNDS(LSP'.TGFLDP'.IDTYPE,LMIN,LMAX);
3403:                IF (LVAL.IVAL > LMAX) OR (LVAL.IVAL < LMIN) THEN ERROR(181);
3404:                LSP1 := LSP'.FSTVAR;
3405:                WHILE LSP1 <> NIL DO
3406:                 WITH LSP1' DO
3407:                  IF VARVAL.IVAL = LVAL.IVAL THEN
3408:                   BEGIN LSP := SUBVAR; LSIZE := SIZE;
3409:                    GOTO 1
3410:                   END
3411:                  ELSE LSP1 := NXTVAR;
3412:                LSIZE := LSP'.SIZE; LSP := NIL
3413:               END
3414:              ELSE ERROR(116);
3415:       1:  END (*WHILE*) ;
3416:        NEWGEN(LSIZE);
3417:        RESETGATTRP(LATTRP);
3418:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3419:       END (*NEWPROC*) ;
3420:
3421:       PROCEDURE MARKRELEASE;
3422:       VAR LATTRP: ATTRP;
3423:       PROCEDURE MARKGEN;
3424:  (**) (*.MARKGEN. GENERATE A #MARK# OPERATION. THE VALUE OF THE HEAP IS TO
3425:         BE SAVED IN THE POINTER VARIABLE DESCRIBED BY #GATTRP'# *) BEGIN
3426:       END (*MARKGEN*);
3427:
3428:       PROCEDURE RELEASEGEN;
3429:  (**) (*.RELEASEGEN. GENERATE A #RELEASE# OPERATION. THE VALUE OF THE HEAP
3430:         HEAP IS TO BE RESET TO THE VALUE OF THE POINTER VARIABLE DESCRIBED BY
3431:         #GATTRP'# *) BEGIN
3432:       END (*RELEASEGEN*);
```

```
3433:
3434:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3435:        LATTRP := GATTRP;
3436:        VARIABLE(FSYS+[COMMA,RPARENT]);
3437:        IF GATTRP'.TYPTR <> NIL THEN
3438:         IF GATTRP'.TYPTR'.FORM = POINTER THEN
3439:         BEGIN
3440:          IF LKEY = 13 THEN
3441:           BEGIN
3442:             MARKGEN;
3443:           END
3444:          ELSE
3445:           BEGIN
3446:            IF DEBUG THEN CHECKPOINTERS(FALSE);
3447:            RELEASEGEN;
3448:            END
3449:         END
3450:         ELSE ERROR(116);
3451:        RESETGATTRP(LATTRP);
3452:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3453:       END (*RELEASE*) ;
3454:
3455:       PROCEDURE ROUNDF;
3456:        PROCEDURE ROUNDGEN;
3457:  (**)  (*.ROUNDGEN. GENERATE A #ROUND# OPERATION ON THE EXPRESSION DES-
3458:          CRIBED BY #GATTRP'#*) BEGIN GATTRP'.TYPTR := INTPTR;
3459:        END (*ROUNDGEN*);
3460:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3461:        EXPRESSION(FSYS+[RPARENT]);
3462:        IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN ROUNDGEN ELSE ERROR(125);
3463:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3464:       END (*ROUNDF*) ;
3465:
3466:       PROCEDURE ABSF;
3467:        PROCEDURE ABSGEN;
3468:  (**)  (*.ABSGEN. GENERATE AN #ABS# OPERATION ON THE EXPRESSION DESCRIBED BY
3469:          #GATTRP'. #GATTRP'.TYPTR# REMAINS UNCHANGED *) BEGIN
3470:        END (*ABSGEN*);
3471:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3472:        EXPRESSION(FSYS+[RPARENT]);
3473:        IF COMPTYPES(GATTRP'.TYPTR,INTPTR)OR COMPTYPES(GATTRP'.TYPTR,
3474:         REALPTR) THEN ABSGEN ELSE ERROR(125);
3475:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3476:       END (*ABSF*) ;
3477:
3478:       PROCEDURE SQRF;
3479:        PROCEDURE SQRGEN;
3480:  (**)  (*.SQRGEN. GENERATE A #SQR# OPERATION ON THE EXPRESSION DESCRIBED BY
3481:          #GATTRP'#. #GATTRP'.TYPTR# REMAINS UNCHANGED*) BEGIN
3482:        END (*SQRGEN*);
3483:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3484:        EXPRESSION(FSYS+[RPARENT]);
3485:        IF COMPTYPES(GATTRP'.TYPTR,INTPTR) OR
3486:            COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN SQRGEN ELSE ERROR(125);
3487:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3488:       END (*SQRF*) ;
3489:
3490:       PROCEDURE TRUNCF;
3491:        PROCEDURE TRUNCGEN;
3492:  (**)  (*.TRUNCGEN. GENERATE A #TRUNC# OPERATION ON THE EXPRESSION DESCRIBED
3493:          BY #GATTRP'# *) BEGIN
3494:         GATTRP'.TYPTR := INTPTR;
3495:        END (*TRUNCGEN*);
3496:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3497:        EXPRESSION(FSYS+[COMMA,RPARENT]);
3498:        IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN TRUNCGEN ELSE ERROR(125);
```

```
3499:         GATTRP'.TYPTR := INTPTR;
3500:         IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3501:       END (*TRUNCF*) ;
3502:
3503:       PROCEDURE ODDP;
3504:         PROCEDURE ODDGEN;
3505:  (**)   (*.ODDGEN. GENERATE AN #ODD# OPERATION ON THE EXPRESSION DESCRIBED BY
3506:            #GATTRP'# *) BEGIN
3507:          GATTRP'.TYPTR := BOOLPTR;
3508:         END (*ODDGEN*);
3509:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3510:         EXPRESSION(FSYS+[RPARENT]);
3511:         IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN ODDGEN ELSE ERROR(125);
3512:         IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3513:       END (*ODDP*) ;
3514:
3515:       PROCEDURE ORDF;
3516:       VAR LSP: STP; LMIN,LMAX: INTEGER;
3517:         PROCEDURE ORDGEN(FSP:STP);
3518:  (**)   (*.ORDGEN. GENERATE AN #ORD# OPERATION ON THE EXPRESSION DESCRIBED BY
3519:           #GATTRP'#. FILL # GATTRP'# WITH THE DESCRIPTION OF THE RESULT.
3520:           #FSP'# IS THE TYPE OF THE INTEGER SUBRANGE CORRESPONDING TO THE TYPE
3521:           OF THE ARGUMENT OF THE FUNCTION #ORD# *) BEGIN
3522:         END (*ORDGEN*);
3523:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3524:         EXPRESSION(FSYS+[RPARENT]);
3525:         IF GATTRP'.TYPTR <> NIL THEN
3526:          IF GATTRP'.TYPTR'.FORM > POWER THEN ERROR(125)
3527:          ELSE
3528:           BEGIN
3529:            NEW(LSP,SUBRANGE); GETBOUNDS(GATTRP'.TYPTR,LMIN,LMAX);
3530:            WITH LSP' DO
3531:             BEGIN FTYPE := FALSE; FORM := SUBRANGE; RANGETYPE := INTPTR;
3532:              MIN.IVAL := LMIN; MAX.IVAL := LMAX; SIZER(LSP);
3533:             END;
3534:            ORDGEN(LSP);
3535:            GATTRP'.TYPTR := LSP;
3536:           END;
3537:         IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3538:       END (*ORDF*) ;
3539:
3540:       PROCEDURE CHRF;
3541:         PROCEDURE CHRGEN;
3542:  (**)   (*.CHRGEN. GENERATE A #CHR# OPERATION ON THE EXPRESSION DESCRIBED BY
3543:          #GATTRP'# *) BEGIN
3544:          GATTRP'.TYPTR := CHARPTR;
3545:         END (*CHRGEN*);
3546:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3547:         EXPRESSION(FSYS+[RPARENT]);
3548:         IF NOT COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN ERROR(125)
3549:         ELSE CHRGEN;
3550:         IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3551:       END (*CHRF*) ;
3552:
3553:       PROCEDURE PREDSUCCF;
3554:
3555:         PROCEDURE SUCCGEN;
3556:  (**)   (*.SUCCGEN. GENERATE A #SUCC# OPERATION ON THE EXPRESSION DESCRIBED BY
3557:          #GATTRP'#. PUT THE RESULT INTO #GATTRP'#.
3558:          #GATTRP'.TYPTR# REMAINS UNCHANGED *) BEGIN
3559:         END (*SUCCGEN*);
3560:
3561:         PROCEDURE PREDGEN;
3562:  (**)   (*.PREDGEN. GENERATE A #PRED# OPERATION ON THE EXPRESSION DESCRIBED
3563:          BY #GATTRP'#. PUT THE RESULT INTO #GATTRP'#.
3564:          #GATTRP'.TYPTR# REMAINS UNCHANGED *) BEGIN
```

```
3565:        END (*PREDGEN*);
3566:
3567:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3568:        EXPRESSION(FSYS+[RPARENT]);
3569:        IF GATTRP'.TYPTR <> NIL THEN
3570:        WITH GATTRP' DO
3571:         IF TYPTR'.FORM > SUBRANGE THEN ERROR(125)
3572:         ELSE
3573:          IF COMPTYPES(TYPTR,REALPTR) THEN ERROR(125);
3574:        IF LKEY = 10 THEN PREDGEN ELSE SUCCGEN;
3575:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3576:       END (*PREDSUCCF*);
3577:
3578:       PROCEDURE STDFLFUNCS;
3579:        PROCEDURE EOLNGEN;
3580:  (**)  (*.EOLNGEN. GENERATE AN #EOLN# OPERATION ON THE FILE DESCRIBED
3581:         BY #GATTRP'#. PUT THE RESULT INTO #GATTRP'# *) BEGIN
3582:         WITH GATTRP' DO BEGIN TYPTR := BOOLPTR; KIND := VARBL; EXPR := TRUE END;
3583:        END (*EOLNGEN*);
3584:
3585:        PROCEDURE EOFGEN;
3586:  (**)  (*.EOFGEN. GENERATE AN #EOF# OPERATION ON THE FILE DESCRIBED BY #GATTRP'#.
3587:          PUT THE RESULT INTO #GATTRP'# *) BEGIN
3588:         WITH GATTRP' DO BEGIN TYPTR := BOOLPTR; KIND := VARBL; EXPR := TRUE END;
3589:        END (*EOFGEN*);
3590:
3591:       BEGIN
3592:        IF SY = LPARENT THEN
3593:         BEGIN INSYMBOL; VARIABLE(FSYS+[RPARENT]);
3594:          IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3595:         END
3596:        ELSE
3597:         STANDFILE(INPUTPTR);
3598:        IF GATTRP'.TYPTR <> NIL THEN
3599:         WITH GATTRP', TYPTR' DO
3600:          IF FORM = FILES THEN
3601:           BEGIN
3602:            IF LKEY = 2 (*EOLN*) THEN
3603:             BEGIN IF NOT TEXTFILE THEN ERROR(125); EOLNGEN;
3604:             END
3605:            ELSE EOFGEN;
3606:           END
3607:          ELSE ERROR(125);
3608:       END (*STDFLFUNCS*);
3609:
3610:       PROCEDURE STDARITHFUNCS;
3611:       TYPE FCTNAME = (SINF,COSF,EXPF,SQRTF,LNF,ARCTANF);
3612:       VAR LFCTN: FCTNAME;
3613:        PROCEDURE STARFCTGEN(FCTN: FCTNAME);
3614:  (**)  (*.STARFCTGEN. GENERATE THE ARITHMETIC FUNCTION GIVEN BY #FCTN#.
3615:         THE ARGUMENT IS DESCRIBED BY #GATTRP'#.
3616:         PUT THE DESCRIPTION OF THE RESULT INTO #GATTRP'# *) BEGIN
3617:        END (*STARFCTGEN*);
3618:       BEGIN IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
3619:        EXPRESSION(FSYS+[RPARENT]);
3620:        IF NOT COMPTYPES(GATTRP'.TYPTR,REALPTR)THEN
3621:         IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN INTTOREAL(GATTRP)
3622:         ELSE ERROR(125);
3623:       CASE LKEY OF
3624:        12: LFCTN := SINF; 13: LFCTN := COSF; 14: LFCTN := EXPF;
3625:        15: LFCTN := SQRTF; 16: LFCTN := LNF; 17: LFCTN := ARCTANF;
3626:       END;
3627:       STARFCTGEN(LFCTN);
3628:        GATTRP'.TYPTR := REALPTR;
3629:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3630:       END (*STDARITHFUNCS*) ;
```

```
3631:
3632:       PROCEDURE CALLNONSTANDARD;
3633:        VAR NXT,LCP: CTP; LSP: STP; LKIND: IDKIND; FTN: BOOLEAN;
3634:          LPARCNT: INTEGER; LATTRP: ATTRP;
3635:          LMIN,LMAX: INTEGER; LP: LBP; FIRST: BOOLEAN; LCHCNT: INTEGER;
3636:          PASS: (VAL,VARADDR,PROCDESC); LCOMP: BOOLEAN;
3637:  (**)    (*.CALLNONSTANDARD. OTHER LOCAL DECLARATIONS*)
3638:
3639:          PROCEDURE PREPCALL;
3640:  (**)    (*.PREPCALL. PREPARE A PROC/FUNC CALL. THIS PROCEDURE IS CALLED BEFORE
3641:            THE PROCESSING OF THE PARAMETERS*) BEGIN
3642:          END (*PREPCALL*);
3643:
3644:          PROCEDURE PREPPARAM(FPARAM: CTP);
3645:  (**)    (*.PREPPARAM. THIS PROCEDURE IS CALLED BEFORE PROCESSING AN
3646:            ACTUAL PARAMETER. #FPARAM# DESCRIBES THE CORRESPONDING FORMAL
3647:            PARAMETER*) BEGIN
3648:          END (*PREPPARAM*);
3649:
3650:          PROCEDURE LOADPROCFUNC(ACTUALPAR,FORMPAR: CTP; FCNT: INTEGER; FORTRAN:
3651:            BOOLEAN);
3652:  (**)     (*.LOADPROCFUNC. GENERATE CODE TO LOAD ONTO THE STACK
3653:              THE ADDRESS (DESCRIPTOR) OF THE PROC/FUNC
3654:              DESCRIBED BY #ACTUALPAR'# (IT IS AN ACTUAL PARAMETER).
3655:              #FORMPAR# GIVES THE FORMAL PARAMETER.
3656:              #FCNT IS THE PARAMETER COUNT.
3657:              #FORTRAN# IS TRUE IFF THE CALLED PROC/FUNC IS A FORTRAN ROUTINE.
3658:              CORRECT #LC# BY THE AMOUNT OF STORE OCCUPIED BY THE PARAMETER*) BEGIN
3659:           SETLCMAX;
3660:          END (*LOADPROCFUNC*);
3661:
3662:          PROCEDURE LOADVALUE(FORMPAR: CTP; FCNT: INTEGER; FORTRAN: BOOLEAN);
3663:  (**)     (*.LOADVALUE. GENERATE CODE TO LOAD ONTO THE STACK
3664:           THE VALUE DESCRIBED BY #GATTRP'#. THIS PROCEDURE IS CALLED FOR
3665:           SHORT VALUE PARAMETERS. FOR OTHER COMMENTS, REFER TO
3666:           #LOADPROCFUNC# *) BEGIN
3667:          SETLCMAX;
3668:         END (*LOADVALUE*);
3669:
3670:         PROCEDURE LOADADDRESS(FORMPAR: CTP; FCNT: INTEGER; FORTRAN: BOOLEAN);
3671:  (**)   (*.LOADADDRESS. GENERATE CODE TO LOAD ONTO THE STACK THE ADDRESS OF THE
3672:            VARIABLE DESCRIBED BY #GATTRP'#. THIS PROCEDURE IS CALLED FO VAR
3673:            PARAMETERS AS WELL AS FOR LONG VALUE PARAMETERS. FOR OTHER COMMENTS
3674:            REFER TO #LODPROCFUNC#*) BEGIN
3675:           SETLCMAX;
3676:         END (*LOADADDRESS*);
3677:
3678:         PROCEDURE CALLGENERATE;
3679:  (**)    (*.CALLGENERATE. GENERATE A CALL OF THE PROCEDURE DESCRIBED BY
3680:            #FCP'# (GLOBAL). THE LEVEL OF THE CALLING PROCEDURE IS GIVEN BY
3681:            #LEVEL#, THE LEVEL OF THE CALLED PROCEDURE IS GIVEN BY #FCP'.PFLEV#
3682:              *)  BEGIN
3683:         END (*CALLGENERATE*);
3684:
3685:        PROCEDURE FCTRESULT;
3686:         (* FILL A NEW #GATTRP'# WITH THE DESCRIPTION OF THE FUNCTION RESULT.
3687:           SET #LC# TO THE VALUE IT MUST HAVE AFTER THE CALL.
3688:           GLOBAL VARIABLES:
3689:           FCP' (GLOBAL)  DESCRIBES THE CALLED PROC/FUNC*)
3690:  (**)   (*.FCTRESULT. LOCAL DECLARATION*)
3691:        BEGIN
3692:         ATTRNEW;
3693:         WITH GATTRP' DO
3694:          BEGIN TYPTR := FCP'.IDTYPE; KIND := VARBL; EXPR := TRUE;
3695:  (**)     (*.FCTRESULT. FILL THE OTHER FIELDS OF #GATTRP'# AND
3696:            SET #LC# *)
```

```
3697:           END
3698:         END (*FCTRESULT*);
3699:
3700:        BEGIN (*CALLNONSTANDARD*)
3701:         WITH FCP' DO
3702:          BEGIN NXT := NEXT; LKIND := PFKIND;
3703:           FTN := FALSE;
3704:           IF LKIND = ACTUAL THEN FTN := PFDECL = FTNDECL;
3705:           PREPCALL;
3706:          END;
3707:         LPARCNT := 0;
3708:         LATTRP := GATTRP;
3709:         IF SY = LPARENT THEN
3710:          BEGIN
3711:           REPEAT PASS := VAL;
3712:            IF LKIND = ACTUAL THEN
3713:             IF NXT = NIL THEN ERROR(126)
3714:             ELSE
3715:              BEGIN
3716:               IF NXT'.KLASS IN [PROC,FUNC] THEN PASS := PROCDESC;
3717:               PREPPARAM(NXT);
3718:              END;
3719:            (*NOTE THAT IN THIS IMPLEMENTATION FORMAL PROC/FUNC MUST
3720:            ONLY HAVE VALUE PARAMETERS*)
3721:            INSYMBOL;
3722:            LPARCNT := LPARCNT + 1;
3723:            IF PASS = PROCDESC THEN
3724:             BEGIN
3725:              IF SY <> IDENT THEN
3726:               BEGIN ERROR(2); SKIP(FSYS+[COMMA,RPARENT]) END
3727:              ELSE
3728:               BEGIN
3729:                LSP := NIL;
3730:                IF NXT'.KLASS = PROC THEN SEARCHID([PROC],LCP)
3731:                ELSE
3732:                 BEGIN SEARCHID([FUNC],LCP);
3733:                  LSP := NXT'.IDTYPE;
3734:                  IF NOT COMPTYPES(LCP'.IDTYPE,LSP)
3735:                   THEN ERROR(128)
3736:                 END;
3737:               IF LCP'.PFDECKIND = STANDARD THEN ERROR(164)
3738:               ELSE
3739:                IF LCP'.PFKIND = ACTUAL THEN
3740:                 BEGIN
3741:                  WITH LCP' DO
3742:                   BEGIN
3743:                    IF FTN AND (PFDECL<>FTNDECL) THEN ERROR(173)
3744:                    ELSE
3745:                     IF NOT FTN AND (PFDECL=FTNDECL) THEN ERROR(174);
3746:                    LOADPROCFUNC(LCP,NXT,LPARCNT,FTN);
3747:                   END;
3748:                  (*MAKE SURE THAT PROC/FUNC HAS ONLY VALUE
3749:                  PARAMS OF FIXED SIZE:*)
3750:                  LCP := LCP'.NEXT;
3751:                  WHILE LCP <> NIL DO
3752:                   WITH LCP' DO
3753:                    BEGIN IF KLASS <> VARS THEN ERROR(170)
3754:                     ELSE
3755:                      IF VKIND = INDRCT THEN ERROR(170);
3756:                     LCP := NEXT
3757:                    END
3758:                 END (*LCP'.PFKIND = ACTUAL*)
3759:                ELSE
3760:                 LOADPROCFUNC(LCP,NXT,LPARCNT,FTN)
3761:               END (*SY = IDENT*);
3762:              GATTRP'.TYPTR := INTPTR;
```

```
3763:             INSYMBOL;
3764:             IF NOT (SY IN FSYS+[COMMA,RPARENT]) THEN
3765:              BEGIN ERROR(6); SKIP(FSYS+[COMMA,RPARENT]) END
3766:            END (*PASS = PROCDESC*)
3767:           ELSE
3768:            BEGIN (*THE PARAMETER IS A VAR OR VALUE PARAMETER*)
3769:             EXPRESSION(FSYS+[COMMA,RPARENT]); LSP := NIL;
3770:             IF GATTRP'.TYPTR <> NIL THEN
3771:              BEGIN
3772:               IF LKIND = ACTUAL THEN
3773:                BEGIN
3774:                 IF NXT <> NIL THEN
3775:                  BEGIN LSP := NXT'.IDTYPE;
3776:                   IF LSP <> NIL THEN
3777:                    BEGIN
3778:                      IF (NXT'.VKIND<>DRCT) OR COPYCOND(LSP) THEN
3779:                       PASS := VARADDR
3780:                    END
3781:                  END
3782:                END
3783:               ELSE
3784:                IF COPYCOND(GATTRP'.TYPTR) THEN
3785:                 PASS := VARADDR;
3786:               IF PASS = VAL THEN
3787:                BEGIN
3788:                 IF (LKIND = ACTUAL) AND (NXT <> NIL) THEN
3789:                  IF COMPTYPES(LSP,REALPTR) THEN
3790:                   BEGIN
3791:                    IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN
3792:                      INTTOREAL(GATTRP)
3793:                   END
3794:                  ELSE
3795:                   IF LSP <> NIL THEN
3796:                    IF DEBUG THEN
3797:                     IF (LSP <> INTPTR) AND (LSP'.FORM <= SUBRANGE) THEN
3798:                      BEGIN GETBOUNDS(LSP,LMIN,LMAX);
3799:                       CHECKBNDS(LMIN,LMAX,2)
3800:                      END
3801:                     ELSE
3802:                      IF LSP'.FORM = POINTER THEN CHECKPOINTER(TRUE);
3803:                END
3804:               ELSE
3805:                BEGIN
3806:                 IF (LKIND = ACTUAL) AND (NXT <> NIL) THEN
3807:                  IF NXT'.VKIND = INDRCT THEN
3808:                   WITH GATTRP' DO
3809:                    IF KIND = CST THEN
3810:                     BEGIN ERROR(154); TYPTR := NIL END
3811:                    ELSE
3812:                     IF EXPR THEN BEGIN ERROR(154); TYPTR := NIL END
3813:                     ELSE
3814:                      BEGIN IF INPCKDSTR THEN ERROR(127);
3815:                       IF FORPOINTER <> NIL THEN ERROR(179);
3816:                      END;
3817:                END;
3818:               IF (LKIND = ACTUAL)AND (NXT <> NIL) THEN
3819:                BEGIN
3820:                 IF NXT'.VKIND = INDRCT THEN
3821:                  LCOMP := NXT'.IDTYPE = GATTRP'.TYPTR
3822:                 ELSE LCOMP := COMPTYPES(NXT'.IDTYPE,GATTRP'.TYPTR);
3823:                END
3824:               ELSE LCOMP := TRUE;
3825:               IF LCOMP THEN
3826:                IF PASS = VAL THEN LOADVALUE(NXT,LPARCNT,FTN)
3827:                ELSE LOADADDRESS(NXT,LPARCNT,FTN)
3828:               ELSE ERROR(142);
```

```
3829:             END (*GATTRP'.TYPTR <> NIL*);
3830:           END (*PASS <> PROCDESC*);
3831:          IF (LKIND = ACTUAL)AND (NXT <> NIL) THEN NXT := NXT'.NEXT
3832:         UNTIL SY <> COMMA;
3833:        IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
3834:       END (*IF LPARENT*);
3835:       CALLGENERATE;
3836:       RESETGATTRP(LATTRP);
3837:       IF FCP'.KLASS = FUNC THEN FCTRESULT;
3838:       LP := FSTLABP; FIRST := TRUE;
3839:       WHILE LP <> NIL DO
3840:        WITH LP' DO
3841:         BEGIN
3842:          IF LCNT <> 0 THEN
3843:           IF LCNT IN FCP'.EXITLAB THEN
3844:            IF DEFINED THEN
3845:             BEGIN
3846:              IF GLOBN < LABCONTR THEN
3847:               BEGIN
3848:                IF FIRST THEN
3849:                 BEGIN ERROR(184); LCHCNT := CHCNT; ENDOFLINE; FIRST := FALSE;
3850:                 END;
3851:                WRITELN(OUTPUT,# LABEL#,LABVAL);
3852:               END;
3853:             END
3854:            ELSE
3855:             IF GLOBN < LABCONTR THEN LABCONTR := GLOBN;
3856:          LP := LP'.NEXTLAB;
3857:         END;
3858:        IF NOT FIRST THEN
3859:         BEGIN CHCNT := LCHCNT;
3860:          FOR LCHCNT := 1 TO CHCNT DO LINE[LCHCNT] := # #;
3861:         END;
3862:       END (*CALLNONSTANDARD*) ;
3863:
3864:      BEGIN (*CALL*)
3865:       IF FCP'.PFDECKIND = STANDARD THEN
3866:        BEGIN
3867:         LKEY := FCP'.KEY;
3868:         IF FCP'.KLASS = PROC THEN
3869:          BEGIN
3870:           CASE LKEY OF
3871:            1,2,   (*GET,PUT*)
3872:            3,4:   (*RESET,REWRITE*)
3873:                   STDFLPROCS;
3874:            5:     PAGE;
3875:            6,7:   (*READ,READLN*)
3876:                   READ;
3877:            8,9:   (*WRITE,WRITELN*)
3878:                   WRITE;
3879:            10:    PACK;
3880:            11:    UNPACK;
3881:            12:    NEWPROC;
3882:            13,14: MARKRELEASE
3883:           END
3884:          END
3885:         ELSE
3886:          BEGIN
3887:           CASE LKEY OF
3888:            1,2:   (*EOF,EOLN*)                    (*PREDICATES*)
3889:                   STDFLFUNCS;
3890:            3:     ODDP;
3891:            4:     ROUNDF;                         (*FUNCTIONS*)
3892:            5:     TRUNCF;
3893:            6:     ABSF;
3894:            7:     SQRF;
```

```
3895:            8:     ORDF;
3896:            9:     CHRF;
3897:            10,11: PREDSUCCF;
3898:            12,13, (*SIN,COS*)
3899:            14,15, (*EXP,SQRT*)
3900:            16,17: (*LN,ARCTAN*)
3901:                   STDARITHFUNCS
3902:           END
3903:          END;
3904:        END (*STANDARD PROCSY AND FUNCTIONS*)
3905:       ELSE CALLNONSTANDARD;
3906:      END (*CALL*) ;
3907:
3908:      PROCEDURE EXPRESSION;
3909:       VAR LATTRP: ATTRP; LOP: OPERATOR;
3910:
3911:       (*THE FOLLOWING PROCEDURES HAVE ONE ARGUMENT. THIS ARGUMENT IS DESCRIBED
3912:         BY #GATTRP'#. THE RESULT HAS TO BE PUT INTO #GATTRP'#, TOO*)
3913:
3914:       PROCEDURE NEGATE;
3915:  (**) (*.NEGATE. GENERATE CREATION OF THE OPPOSITE OF REAL OR INTEGER ARGUMENT.
3916:         #GATTRP'.TYPTR# REMAINS UNCHANGED *) BEGIN
3917:       END (*NEGATE*);
3918:
3919:       PROCEDURE NOTFACTOR;
3920:  (**) (*.NOTFACTOR. GENERATE NEGATION OPERATION FOR BOOLEAN ARGUMENT.
3921:         #GATTR'.TYPTR# REMAINS UNCHANGED*) BEGIN
3922:       END (*NOTFACTOR*);
3923:
3924:       PROCEDURE PREPFIRSTOPD(FOP: OPERATOR);
3925:  (**) (*.PREPFIRSTOPD. PREPARE THE FIRST OPERAND DESCRIBED BY #GATTRP'# FOR THE
3926:         OPERATION DESCRIBED BY THE OPERATOR #FOP#*) BEGIN
3927:       END (*PREPFIRSTOPD*);
3928:
3929:       (*THE FOLLOWING PROCEDURES HAVE TWO ARGUMENTS. THE FIRST ONE IS DESCRIBED
3930:         BY THE PARAMETER #FATTRP'# (JUST BELOW THE TOP OF THE ATTRIBUTE STACK),
3931:         THE OTHER ONE IS DESCRIBED BY #GATTRP'#. THE DESCRIPTION OF THE RESULT
3932:         IS TO BE PUT INTO #FATTRP'#*)
3933:
3934:       PROCEDURE INTARITH(FATTRP: ATTRP; FOP: OPERATOR);
3935:  (**)  (*.INTARITH. GENERATE THE OPERATION #FOP# ON THE ARGUMENTS (BOTH
3936:          INTEGERS). #FOP# IS IN [PLUS,MINUS,MUL,IDIV,IMOD] *) BEGIN
3937:        GATTRP'.TYPTR := INTPTR;
3938:       END (*INTARITH*);
3939:
3940:       PROCEDURE REALARITH(FATTRP: ATTRP; FOP: OPERATOR);
3941:  (**)  (*.REALARITH. GENERATE OPERATION GIVING A REAL RESULT.
3942:          #FOP# IS IN [PLUS,MINUS,MUL,RDIV];
3943:          PLUS,MINUS,MUL: ONE OF THE ARGUMENTS IS REAL
3944:          THE OTHER ONE IS INTEGER OR REAL.
3945:          RDIV: BOTH ARGUMENTS ARE INTEGER OR REAL*) BEGIN
3946:        GATTRP'.TYPTR := REALPTR
3947:       END (*REALARITH*);
3948:
3949:       PROCEDURE SETARITH(FATTRP: ATTRP; FOP: OPERATOR);
3950:  (**) (*.SETARITH. GENERATE OPERATION FOR SETS.
3951:         #FOP# IS IN [PLUS,MINUS,MUL];
3952:         #GATTRP'.TYPTR# REMAINS UNCHANGED *) BEGIN
3953:       END (*SETARITH*);
3954:
3955:       PROCEDURE BOOLARITH(FATTRP: ATTRP; FOP: OPERATOR);
3956:  (**)  (*.BOOLARITH. GENERATE OPERATION FOR BOOLEAN ARGUMENTS.
3957:          #FOP# IS IN [ANDOP,OROP].
3958:          #GATTRP'.TYPTR# REMAINS UNCHANGED*) BEGIN
3959:       END (*BOOLARITH*);
3960:
```

```
3961:       PROCEDURE RELINT(FATTRP: ATTRP; FOP: OPERATOR);
3962:  (**)  (*.RELINT. GENERATE COMPARISON FOR INTEGER ARGUMENTS.
3963:          #FOP# IS IN [NEOP,EQOP,LTOP,LEOP,GTOP,GEOP] *) BEGIN
3964:        FATTRP'.TYPTR := BOOLPTR
3965:       END (*RELINT*);
3966:
3967:       PROCEDURE RELREAL(FATTRP: ATTRP; FOP: OPERATOR);
3968:  (**)  (*.RELREAL. SAME AS #RELINT#, EXCEPT THAT ONE ARGUMENT IS REAL, THE OTHER
3969:          ONE REAL OR INTEGER*) BEGIN
3970:        FATTRP'.TYPTR := BOOLPTR
3971:       END (*RELREAL*);
3972:
3973:       PROCEDURE RELBOOL(FATTRP: ATTRP; FOP: OPERATOR);
3974:  (**) (*.RELBOOL. GENERATE COMPARISON FOR BOOLEAN TYPES.
3975:         THE ARGUMENTS ARE OF THE SAME TYPE*) BEGIN
3976:       END (*RELBOOL*);
3977:
3978:       PROCEDURE RELSIMPLE(FATTRP: ATTRP; FOP: OPERATOR);
3979:  (**)  (*.RELSIMPLE. GENERATE COMPARISON FOR SIMPLE, NONSTANDARD TYPES. THE
3980:          ARGUMENTS ARE OF THE SAME TYPE.
3981:          #FOP# IS LIKE IN #RELINT#*) BEGIN
3982:        FATTRP'.TYPTR := BOOLPTR
3983:       END (*RELSIMPLE*);
3984:
3985:       PROCEDURE INPOWER(FATTRP: ATTRP);
3986:  (**)  (*.INPOWER. GENERATE THE #IN# OPERATION.
3987:          THE ATTRIBUTE JUST BELOW THE TOP OF THE ATTRIBUTE STACK (#FATTRP'#)
3988:          DESCRIBES THE VALUE WHICH MUST BE CHECKED AS BELONGING
3989:          TO THE SET DESCRIBED BY #GATTRP'#. NO TYPE CHECK IS NECESSARY
3990:          IN THIS ROUTINE*) BEGIN
3991:        FATTRP'.TYPTR := BOOLPTR
3992:       END (*INPOWER*);
3993:
3994:       PROCEDURE RELPOWER(FATTRP: ATTRP; FOP: OPERATOR);
3995:  (**)  (*.RELPOWER. GENERATE COMPARISON BETWEEN SETS.
3996:          #FOP# IS IN [EQOP,NEOP,GEOP,LEOP] *) BEGIN
3997:        FATTRP'.TYPTR := BOOLPTR
3998:       END (*RELPOWER*);
3999:
4000:       PROCEDURE RELPOINTER(FATTRP: ATTRP; FOP: OPERATOR);
4001:  (**)  (*.RELPOINTER. GENERATE COMPARISON BETWEEN POINTERS.
4002:          #FOP# IS IN [EQOP,NEOP] *) BEGIN
4003:        FATTRP'.TYPTR := BOOLPTR
4004:       END (*RELPOINTER*);
4005:
4006:       PROCEDURE RELSTRING(FATTRP: ATTRP; FOP: OPERATOR);
4007:  (**)  (*.RELSTRING. GENERATE COMPARISON BETWEEN STRINGS.
4008:          #FOP# IS IN [EQOP,NEOP,LTOP,GTOP,LEOP,GEOP] *) BEGIN
4009:        FATTRP'.TYPTR := BOOLPTR
4010:       END (*RELSTRING*);
4011:
4012:       PROCEDURE SIMPLEEXPRESSION(FSYS: SETOFSYS);
4013:        VAR LATTRP: ATTRP; LOP: OPERATOR; SIGN: (PLUSSIGN,MINUSSIGN,NOSIGN);
4014:
4015:        PROCEDURE TERM(FSYS: SETOFSYS);
4016:         VAR LATTRP: ATTRP; LOP: OPERATOR;
4017:
4018:         PROCEDURE FACTOR(FSYS: SETOFSYS);
4019:          VAR LCP: CTP; LSP,LSP1: STP; LCSTATTRP,LVARATTRP,LATTRP: ATTRP;
4020:            EXITLOOP,GOON: BOOLEAN; N: INTEGER;
4021:
4022:          PROCEDURE SETSUBRANGE(FATTRP: ATTRP; FSP: STP);
4023:  (**)    (*.SETSUBRANGE. LOCAL DECLARATIONS*)
4024:          BEGIN
4025:  (**)     (*.SETSUBRANGE. GENERATE THE SET OF ALL VALUES BETWEEN THE EXPRESSION
4026:             DESCRIBED BY #FATTRP'# AND THAT DESCRIBED BY #GATTRP'# (ONE OF THEM
```

```
4027:             IS NOT CONSTANT).
4028:             #FATTRP'# IS THE ATTRIBUTE JUST BELOW THE TOP OF THE STACK.
4029:             PUT THE DESCRIPTION INTO #FATTRP'#.
4030:             #FSP# POINTS TO THE TYPE OF THE SET*)
4031:           WITH FATTRP' DO
4032:            BEGIN TYPTR := FSP ; KIND := VARBL; EXPR := TRUE END;
4033:          END (*SETSUBRANGE*);
4034:
4035:          PROCEDURE SETELEMENT(FSP:STP);
4036:  (**)    (*.SETELEMENT. LOCAL DECLARATIONS*)
4037:          BEGIN
4038:  (**)     (*.SETELEMENT. GENERATE THE SET OF THE UNIQUE VALUE DESCRIBED BY
4039:             #GATTRP'#. (THIS EXPRESSION IS NOT A CONSTANT).
4040:             PUT THE DESCRIPTION OF THE RESULT IN #GATTRP'#.
4041:             #FSP# POINTS TO THE TYPE OF THE SET *)
4042:           WITH GATTRP' DO
4043:            BEGIN TYPTR := FSP; KIND := VARBL; EXPR := TRUE END;
4044:          END (*SETELEMENT*);
4045:
4046:         BEGIN (*FACTOR*)
4047:          IF NOT (SY IN FACBEGSYS) THEN
4048:           BEGIN ERROR(58); SKIP(FSYS+FACBEGSYS);
4049:            GATTRP'.TYPTR := NIL
4050:           END;
4051:          REPEAT
4052:           IF SY IN FACBEGSYS THEN
4053:           BEGIN
4054:            CASE SY OF
4055:   (*ID*)    IDENT:
4056:              BEGIN SEARCHID([KONST,VARS,FIELD,FUNC],LCP);
4057:               INSYMBOL;
4058:               CASE LCP'.KLASS OF
4059:                KONST:
4060:                 BEGIN ATTRNEW;
4061:                  WITH LCP', GATTRP' DO
4062:                   BEGIN TYPTR := IDTYPE; KIND := CST;
4063:                    CVAL := VALUES;
4064:  (**)              (*.FACTCSTID. INITIALIZE MACHINE DEPENDENT INFO*)
4065:                   END;
4066:                  END;
4067:                VARS,
4068:                FIELD:
4069:                 SELECTOR(FSYS,LCP);
4070:                FUNC:
4071:                 CALL(FSYS,LCP)
4072:              END
4073:             END;
4074:   (*CST*)   INTCONST:
4075:              BEGIN ATTRNEW;
4076:               WITH GATTRP' DO
4077:                BEGIN TYPTR := INTPTR; KIND := CST;
4078:                 CVAL.IVAL := IVAL;
4079:  (**)           (*.FACTCSTINT. INITIALIZE MACHINE DEPENDENT INFO*)
4080:                END;
4081:               INSYMBOL
4082:              END;
4083:             REALCONST:
4084:              BEGIN ATTRNEW;
4085:               WITH GATTRP' DO
4086:                BEGIN TYPTR := REALPTR; KIND := CST;
4087:                 CVAL.RVAL := RVAL;
4088:  (**)           (*.FACTCSTREAL. INITIALIZE MACHINE DEPENDENT INFO*)
4089:                END;
4090:               INSYMBOL
4091:              END;
4092:             CHARCONST:
```

```
4093:              BEGIN ATTRNEW;
4094:               WITH GATTRP' DO
4095:                BEGIN TYPTR := CHARPTR; KIND := CST;
4096:                 CVAL.IVAL := IVAL;
4097:  (**)            (*.FACTCSTCHAR. INITIALIZE MACHINE DEPENDENT INFO*)
4098:                END;
4099:               INSYMBOL
4100:              END;
4101:             STRINGCONST:
4102:              BEGIN ATTRNEW;
4103:               WITH GATTRP' DO
4104:                BEGIN STRINGTYPE(TYPTR); KIND := CST;
4105:                 CVAL.SVAL := SVAL;
4106:  (**)            (*.FACTCSTSTRING. INITIALIZE MACHINE DEPENDENT INFO*)
4107:                END;
4108:               INSYMBOL
4109:              END;
4110:    (*(*)    LPARENT:
4111:              BEGIN INSYMBOL; EXPRESSION(FSYS+[RPARENT]);
4112:               IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
4113:              END;
4114:   (*NOT*)  NOTSY:
4115:              BEGIN INSYMBOL; FACTOR(FSYS);
4116:               IF COMPTYPES(GATTRP'.TYPTR,BOOLPTR) AND
4117:                (GATTRP'.TYPTR <> NIL) THEN NOTFACTOR
4118:               ELSE
4119:                BEGIN ERROR(135); GATTRP'.TYPTR := NIL END
4120:              END;
4121:    (*[*)     LBRACK:
4122:              BEGIN INSYMBOL;
4123:               NEW(LSP,POWER);
4124:               WITH LSP' DO
4125:                BEGIN ELSET := NIL; FORM := POWER; PCKDSET := FALSE;
4126:                 FTYPE := FALSE;
4127:                 SIZER(LSP)
4128:                END;
4129:               ATTRNEW; LCSTATTRP := GATTRP; LVARATTRP := NIL;
4130:               WITH LCSTATTRP' DO
4131:                BEGIN TYPTR := LSP; KIND := CST; CVAL.PVAL := [ ];
4132:  (**)            (*.FACTCSTSET. INITIALIZE MACHINE DEPENDENT INFO*)
4133:                END;
4134:               IF SY = RBRACK THEN INSYMBOL
4135:               ELSE
4136:                BEGIN
4137:                 (*LOOP UNTIL SY <> COMMA:*)
4138:                 REPEAT
4139:                  EXPRESSION(FSYS+[COMMA,COLON,RBRACK]);
4140:                  IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN
4141:                   BEGIN ERROR(109); GATTRP'.TYPTR := NIL END;
4142:                  IF GATTRP'.TYPTR <> NIL THEN
4143:                   IF GATTRP'.TYPTR'.FORM > SUBRANGE THEN
4144:                    BEGIN ERROR(136); GATTRP'.TYPTR := NIL END
4145:                   ELSE
4146:                    IF COMPTYPES(LSP'.ELSET,GATTRP'.TYPTR) THEN
4147:                     BEGIN
4148:                      IF LSP'.ELSET = NIL THEN
4149:                       BEGIN LSP1 := GATTRP'.TYPTR;
4150:                        IF LSP1 <> NIL THEN
4151:                         BEGIN GOON := LSP1'.FORM = SUBRANGE;
4152:                          WHILE GOON DO
4153:                           BEGIN LSP1 := LSP1'.RANGETYPE;
4154:                            IF LSP1 <> NIL THEN GOON := LSP1'.FORM = SUBRANGE
4155:                            ELSE GOON := FALSE;
4156:                           END;
4157:                         END;
4158:                       LSP'.ELSET := LSP1;
```

```
4159:                     END;
4160:                   IF GATTRP'.KIND = CST THEN
4161:                    BEGIN
4162:                     IF (GATTRP'.CVAL.IVAL<SETMIN) OR
4163:                       (GATTRP'.CVAL.IVAL>SETMAX) THEN
4164:                      BEGIN ERROR(304); GATTRP'.CVAL.IVAL := SETMIN END;
4165:                    END
4166:                   ELSE
4167:                    IF DEBUG THEN CHECKBNDS(SETMIN,SETMAX,2);
4168:                  END
4169:                ELSE BEGIN ERROR(137); LSP'.ELSET := NIL END;
4170:              IF SY = COLON THEN
4171:               BEGIN LATTRP := GATTRP;
4172:                INSYMBOL;
4173:                EXPRESSION(FSYS+[COMMA,RBRACK]);
4174:                IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN
4175:                 BEGIN ERROR(109); GATTRP'.TYPTR := NIL END;
4176:                IF GATTRP'.TYPTR <> NIL THEN
4177:                 IF GATTRP'.TYPTR'.FORM > SUBRANGE THEN
4178:                  BEGIN ERROR(136); GATTRP'.TYPTR := NIL
4179:                  END
4180:                 ELSE
4181:                  IF NOT COMPTYPES(LATTRP'.TYPTR,GATTRP'.TYPTR)
4182:                    THEN ERROR(137)
4183:                  ELSE
4184:                   BEGIN
4185:                    IF GATTRP'.KIND = CST THEN
4186:                     BEGIN
4187:                      IF (GATTRP'.CVAL.IVAL<SETMIN) OR
4188:                        (GATTRP'.CVAL.IVAL>SETMAX) THEN
4189:                       BEGIN ERROR(304); GATTRP'.CVAL.IVAL := SETMIN END;
4190:                     END
4191:                    ELSE
4192:                     IF DEBUG THEN CHECKBNDS(SETMIN,SETMAX,2);
4193:                   END;
4194:                IF (LATTRP'.TYPTR <> NIL)AND (GATTRP'.TYPTR <> NIL)
4195:                 THEN
4196:                 BEGIN
4197:                  IF (LATTRP'.KIND = CST) AND (GATTRP'.KIND = CST) THEN
4198:                   BEGIN
4199:                     FOR N := LATTRP'.CVAL.IVAL TO GATTRP'.CVAL.IVAL DO
4200:                      LCSTATTRP'.CVAL.PVAL := LCSTATTRP'.CVAL.PVAL+[N];
4201:                     IF LVARATTRP = NIL THEN RESETGATTRP(LCSTATTRP)
4202:                     ELSE RESETGATTRP(LVARATTRP);
4203:                    END
4204:                  ELSE
4205:                   BEGIN
4206:                    SETSUBRANGE(LATTRP,LSP); RESETGATTRP(LATTRP);
4207:                    IF LVARATTRP <> NIL THEN
4208:                     BEGIN SETARITH(LVARATTRP,PLUS);RESETGATTRP(LVARATTRP) END
4209:                    ELSE LVARATTRP := GATTRP;
4210:                   END
4211:                 END;
4212:               END (*COLON*)
4213:             ELSE (*SY <> COLON*)
4214:               IF GATTRP'.TYPTR <> NIL THEN
4215:                IF GATTRP'.KIND = CST THEN
4216:                 BEGIN
4217:                  LCSTATTRP'.CVAL.PVAL := LCSTATTRP'.CVAL.PVAL
4218:                   + [GATTRP'.CVAL.IVAL];
4219:                 IF LVARATTRP = NIL THEN RESETGATTRP(LCSTATTRP)
4220:                 ELSE RESETGATTRP(LVARATTRP);
4221:                END
4222:               ELSE
4223:                BEGIN
4224:                 SETELEMENT(LSP);
```

```
4225:                     IF LVARATTRP = NIL THEN LVARATTRP := GATTRP
4226:                     ELSE SETARITH(LVARATTRP,PLUS); RESETGATTRP(LVARATTRP) ;
4227:                    END;
4228:                  EXITLOOP := SY <> COMMA;
4229:                  IF NOT EXITLOOP THEN INSYMBOL
4230:                 UNTIL EXITLOOP;
4231:                  IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
4232:                 END;
4233:                IF LVARATTRP <> NIL THEN
4234:                 IF LCSTATTRP'.CVAL.PVAL <> [ ] THEN SETARITH(LCSTATTRP,PLUS)
4235:                 ELSE MOVATTR;
4236:                RESETGATTRP(LCSTATTRP);
4237:               END
4238:             END (*CASE*) ;
4239:             IF NOT (SY IN FSYS) THEN
4240:              BEGIN ERROR(6); SKIP(FSYS+FACBEGSYS) END
4241:            END (*IF*)
4242:           UNTIL SY IN FSYS;
4243:           IF P THEN
4244:            BEGIN
4245:             IF P THEN BEGIN WRITELN(OUTPUT,# FACTOR#); PRINTATTR(GATTRP) END;
4246:            END;
4247:          END (*FACTOR*) ;
4248:
4249:         BEGIN (*TERM*)
4250:          FACTOR(FSYS+[MULOP]);
4251:          WHILE SY = MULOP DO
4252:           BEGIN
4253:            PREPFIRSTOPD(OP);
4254:            LATTRP := GATTRP;
4255:            LOP := OP;
4256:            INSYMBOL; FACTOR(FSYS+[MULOP]);
4257:            IF (LATTRP'.TYPTR <> NIL)AND (GATTRP'.TYPTR <> NIL) THEN
4258:             CASE LOP OF
4259:   (***)        MUL: BEGIN
4260:                      IF NOT COMPTYPES(LATTRP'.TYPTR,GATTRP'.TYPTR) THEN
4261:                       IF COMPTYPES(LATTRP'.TYPTR,INTPTR) AND
4262:                        COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN INTTOREAL(LATTRP)
4263:                       ELSE
4264:                        IF COMPTYPES(GATTRP'.TYPTR,INTPTR) AND
4265:                         COMPTYPES(LATTRP'.TYPTR,REALPTR) THEN INTTOREAL(GATTRP);
4266:                      IF COMPTYPES(LATTRP'.TYPTR,GATTRP'.TYPTR) THEN
4267:                       IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN INTARITH(LATTRP,MUL)
4268:                       ELSE
4269:                        IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN
4270:                         REALARITH(LATTRP,MUL)
4271:                        ELSE
4272:                         IF GATTRP'.TYPTR'.FORM = POWER THEN SETARITH(LATTRP,MUL)
4273:                         ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END
4274:                      ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END;
4275:                     END;
4276:   (*/*)        RDIV: BEGIN
4277:                      IF COMPTYPES(LATTRP'.TYPTR,INTPTR) THEN INTTOREAL(LATTRP);
4278:                      IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN INTTOREAL(GATTRP);
4279:                      IF COMPTYPES(GATTRP'.TYPTR,REALPTR)
4280:                       AND COMPTYPES(LATTRP'.TYPTR,REALPTR) THEN
4281:                        REALARITH(LATTRP,RDIV)
4282:                      ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END;
4283:                     END;
4284:   (*DIV,MOD*) IDIV,IMOD: IF COMPTYPES(LATTRP'.TYPTR,INTPTR) AND
4285:                             COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN
4286:                            INTARITH(LATTRP,LOP)
4287:                           ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END;
4288:   (*AND*)     ANDOP:IF COMPTYPES(LATTRP'.TYPTR,BOOLPTR)AND
4289:                       COMPTYPES(GATTRP'.TYPTR,BOOLPTR) THEN
4290:                      BOOLARITH(LATTRP,ANDOP)
```

```
4291:                   ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END
4292:            END (*CASE*)
4293:           ELSE GATTRP'.TYPTR := NIL;
4294:           RESETGATTRP(LATTRP);
4295:          END (*WHILE*);
4296:         IF P THEN
4297:          BEGIN
4298:           IF P THEN BEGIN WRITELN(OUTPUT,# TERM#); PRINTATTR(GATTRP) END;
4299:          END;
4300:        END (*TERM*) ;
4301:
4302:       BEGIN (*SIMPLEEXPRESSION*)
4303:        SIGN := NOSIGN;
4304:        IF OP IN [PLUS,MINUS] THEN
4305:         BEGIN
4306:          IF OP = PLUS THEN SIGN := PLUSSIGN ELSE SIGN := MINUSSIGN;
4307:          INSYMBOL;
4308:         END;
4309:        TERM(FSYS+[ADDOP]);
4310:        IF SIGN = MINUSSIGN THEN
4311:         IF COMPTYPES(GATTRP'.TYPTR,INTPTR)OR COMPTYPES(GATTRP'.TYPTR,
4312:          REALPTR) THEN NEGATE
4313:         ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END
4314:        ELSE
4315:         IF SIGN = PLUSSIGN THEN
4316:          IF NOT COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN
4317:           IF NOT COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN
4318:            BEGIN ERROR(134); GATTRP'.TYPTR := NIL END;
4319:        WHILE SY = ADDOP DO
4320:         BEGIN
4321:          PREPFIRSTOPD(OP);
4322:          LATTRP := GATTRP; LOP := OP;
4323:          INSYMBOL; TERM(FSYS+[ADDOP]);
4324:          IF (LATTRP'.TYPTR <> NIL)AND (GATTRP'.TYPTR <> NIL) THEN
4325:           CASE LOP OF
4326:  (*+,-*)    PLUS,MINUS:
4327:             BEGIN
4328:              IF NOT COMPTYPES(GATTRP'.TYPTR,LATTRP'.TYPTR) THEN
4329:               IF COMPTYPES(GATTRP'.TYPTR,REALPTR) AND
4330:                COMPTYPES(LATTRP'.TYPTR,INTPTR) THEN INTTOREAL(LATTRP)
4331:               ELSE
4332:                IF COMPTYPES(GATTRP'.TYPTR,INTPTR) AND
4333:                 COMPTYPES(LATTRP'.TYPTR,REALPTR) THEN INTTOREAL(GATTRP);
4334:              IF COMPTYPES(GATTRP'.TYPTR,LATTRP'.TYPTR) THEN
4335:               IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN INTARITH(LATTRP,LOP)
4336:               ELSE
4337:                IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN REALARITH(LATTRP,LOP)
4338:                ELSE
4339:                 IF GATTRP'.TYPTR'.FORM = POWER THEN SETARITH(LATTRP,LOP)
4340:                 ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END
4341:              ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END;
4342:             END;
4343:  (*OR*)     OROP:
4344:             IF COMPTYPES(LATTRP'.TYPTR,BOOLPTR)AND
4345:               COMPTYPES(GATTRP'.TYPTR,BOOLPTR) THEN BOOLARITH(LATTRP,OROP)
4346:             ELSE BEGIN ERROR(134); GATTRP'.TYPTR := NIL END
4347:           END (*CASE*)
4348:          ELSE GATTRP'.TYPTR := NIL;
4349:          RESETGATTRP(LATTRP);
4350:         END (*WHILE*);
4351:        IF P THEN
4352:         BEGIN
4353:          IF P THEN
4354:           BEGIN WRITELN(OUTPUT,# SIMPLEEXPRESSION#); PRINTATTR(GATTRP) END;
4355:         END;
4356:       END (*SIMPLEEXPRESSION*) ;
```

```
4357:
4358:      BEGIN (*EXPRESSION*)
4359:       SIMPLEEXPRESSION(FSYS+[RELOP]);
4360:       IF SY = RELOP THEN
4361:        BEGIN
4362:         PREPFIRSTOPD(OP);
4363:         LATTRP := GATTRP; LOP := OP;
4364:         INSYMBOL; SIMPLEEXPRESSION(FSYS);
4365:         IF (LATTRP'.TYPTR <> NIL)AND (GATTRP'.TYPTR <> NIL) THEN
4366:          IF LOP = INOP THEN
4367:           IF GATTRP'.TYPTR'.FORM = POWER THEN
4368:            IF COMPTYPES(LATTRP'.TYPTR,GATTRP'.TYPTR'.ELSET) THEN
4369:             INPOWER(LATTRP)
4370:            ELSE BEGIN ERROR(129); GATTRP'.TYPTR := NIL END
4371:           ELSE BEGIN ERROR(130); GATTRP'.TYPTR := NIL END
4372:          ELSE
4373:           BEGIN
4374:            IF NOT COMPTYPES(LATTRP'.TYPTR,GATTRP'.TYPTR) THEN
4375:             BEGIN
4376:              IF COMPTYPES(GATTRP'.TYPTR,REALPTR) AND
4377:               COMPTYPES(LATTRP'.TYPTR,INTPTR) THEN INTTOREAL(LATTRP)
4378:              ELSE
4379:               IF COMPTYPES(LATTRP'.TYPTR,REALPTR) AND
4380:                COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN INTTOREAL(GATTRP);
4381:             END;
4382:            IF COMPTYPES(GATTRP'.TYPTR,LATTRP'.TYPTR) THEN
4383:             CASE LATTRP'.TYPTR'.FORM OF
4384:              SCALAR,SUBRANGE:
4385:               BEGIN
4386:                IF COMPTYPES(LATTRP'.TYPTR,REALPTR) THEN RELREAL(LATTRP,LOP)
4387:                ELSE
4388:                 IF COMPTYPES(LATTRP'.TYPTR,INTPTR) THEN RELINT(LATTRP,LOP)
4389:                 ELSE
4390:                  IF COMPTYPES(LATTRP'.TYPTR,BOOLPTR) THEN RELBOOL(LATTRP,LOP)
4391:                  ELSE RELSIMPLE(LATTRP,LOP);
4392:               END;
4393:              POINTER:
4394:               BEGIN
4395:                IF LOP IN [EQOP,NEOP] THEN RELPOINTER(LATTRP,LOP)
4396:                ELSE BEGIN ERROR(131); GATTRP'.TYPTR := NIL END
4397:               END;
4398:              POWER:
4399:               BEGIN
4400:                IF LOP IN [LTOP,GTOP] THEN
4401:                 BEGIN ERROR(132); GATTRP'.TYPTR := NIL END
4402:                ELSE RELPOWER(LATTRP,LOP)
4403:               END;
4404:              ARRAYS:
4405:               BEGIN
4406:                IF STRING(LATTRP'.TYPTR) THEN RELSTRING(LATTRP,LOP)
4407:                ELSE BEGIN ERROR(399); GATTRP'.TYPTR := NIL END
4408:               END;
4409:              RECORDS:
4410:               BEGIN ERROR(399); GATTRP'.TYPTR := NIL END;
4411:              FILES:
4412:               BEGIN ERROR(133); GATTRP'.TYPTR := NIL END
4413:             END (*CASE*)
4414:            ELSE BEGIN ERROR(129); GATTRP'.TYPTR := NIL END;
4415:           END (*SY <> INOP*)
4416:          ELSE GATTRP'.TYPTR := NIL;
4417:         RESETGATTRP(LATTRP);
4418:        END (*SY = RELOP*) ;
4419:       IF P THEN
4420:        BEGIN
4421:         IF P THEN BEGIN WRITELN(OUTPUT,# EXPRESSION#); PRINTATTR(GATTRP) END;
4422:        END;
```

```
4423:      END (*EXPRESSION*);
4424:
4425:      PROCEDURE ASSIGNMENT(FCP: CTP);
4426:       VAR LEFTP,LATTRP: ATTRP; LMIN,LMAX: INTEGER;
4427:  (**)  (*.ASSIGNDECL. OTHER LOCAL DECLARATIONS*)
4428:
4429:       PROCEDURE ASSIGNSYMBOL;
4430:  (**) (*.ASSIGNSYMBOL. THIS PROCEDURE IS CALLED WHEN THE ASSIGNMENT
4431:        SYMBOL IS DETECTED *) BEGIN
4432:       END (*ASSIGNSYMBOL*);
4433:
4434:       PROCEDURE ASSIGN;
4435:  (**)  (*.ASSIGN. ASSIGN THE VALUE OF THE EXPRESSION DESCRIBED BY #GATTRP'# TO
4436:          THE VARIABLE OR FIELD DESCRIBED BY #LEFTP'#. THE TYPES
4437:          ARE COMPATIBLE*) BEGIN
4438:       END (*ASSIGN*);
4439:
4440:      BEGIN
4441:       LATTRP := GATTRP;
4442:       SELECTOR(FSYS+[BECOMES],FCP);
4443:       IF SY = BECOMES THEN
4444:        BEGIN
4445:         ASSIGNSYMBOL;
4446:         LEFTP := GATTRP;
4447:         IF GATTRP'.FORPOINTER <> NIL THEN ERROR(177);
4448:         INSYMBOL; EXPRESSION(FSYS);
4449:         IF (LEFTP'.TYPTR <> NIL) AND (GATTRP'.TYPTR <> NIL) THEN
4450:          BEGIN
4451:           IF COMPTYPES(LEFTP'.TYPTR,REALPTR) THEN
4452:            BEGIN
4453:             IF COMPTYPES(GATTRP'.TYPTR,INTPTR) THEN
4454:              BEGIN INTTOREAL(GATTRP); ASSIGN END
4455:             ELSE
4456:              IF COMPTYPES(GATTRP'.TYPTR,REALPTR) THEN ASSIGN
4457:              ELSE ERROR(129)
4458:            END
4459:           ELSE
4460:            IF COMPTYPES(GATTRP'.TYPTR,LEFTP'.TYPTR) THEN
4461:             CASE LEFTP'.TYPTR'.FORM OF
4462:              SCALAR,
4463:              SUBRANGE:
4464:               BEGIN
4465:                IF LEFTP'.TYPTR <> INTPTR THEN
4466:                 BEGIN
4467:                  GETBOUNDS(LEFTP'.TYPTR,LMIN,LMAX);
4468:                  IF GATTRP'.KIND = CST THEN
4469:                   BEGIN IF (GATTRP'.CVAL.IVAL<LMIN) OR (GATTRP'.CVAL.IVAL>LMAX)
4470:                          THEN ERROR(303)
4471:                   END
4472:                   ELSE
4473:                   IF DEBUG THEN CHECKBNDS(LMIN,LMAX,2)
4474:                 END;
4475:                ASSIGN
4476:               END;
4477:              POINTER:
4478:               BEGIN IF DEBUG THEN CHECKPOINTER(TRUE); ASSIGN END;
4479:              POWER: ASSIGN;
4480:              ARRAYS,
4481:              RECORDS: IF LEFTP'.TYPTR'.FTYPE THEN ERROR(146)
4482:                       ELSE ASSIGN;
4483:              FILES: ERROR(146)
4484:             END
4485:            ELSE ERROR(129)
4486:          END;
4487:        END (*SY = BECOMES*)
4488:       ELSE ERROR(51);
```

```
4489:       RESETGATTRP(LATTRP);
4490:      END (*ASSIGNMENT*) ;
4491:
4492:      PROCEDURE GOTOSTATEMENT;
4493:       LABEL 1;
4494:       VAR LLP: LBP; I: LEVRANGE; LFSTOCC: LOCOFREF; LUSF: USFREF;
4495:
4496:       PROCEDURE GOTODEF(FLP: LBP);
4497:  (**) (*.GOTODEF.THIS PROCEDURE IS CALLED UPON DETECTION OF A LABEL WHICH
4498:        IS ALREADY DEFINED. GENERATE A JUMP TO THE LABEL DESCIBED BY
4499:        #FLP'# *) BEGIN
4500:        GENJMP(FLP'.LABADDR);
4501:       END (*GOTODEF*);
4502:
4503:       PROCEDURE GOTONOTDEF(FLP: LBP);
4504:  (**) (*.GOTONOTDEF. THIS PROCEDURE IS CALLED UPON DETECTION OF A LOCAL LABEL
4505:         WHICH IS NOT YET DEFINED. PREPARE GENERATION OF A JUMP TO THE
4506:         LABEL DESCRIBED BY #FLP'# *) BEGIN
4507:        WITH FLP' DO
4508:         BEGIN PREPJMP(LUSF); LFSTOCC := FSTOCC;
4509:          LINKOCC(LFSTOCC,LUSF); FSTOCC := LFSTOCC;
4510:         END;
4511:       END (*GOTONOTDEF*);
4512:
4513:       PROCEDURE GENLGJMP(FLP: LBP);
4514:  (**) (*.GENLGJMP. GENERATE A LONG JUMP (OUTSIDE OF CURRENT PROC/FUNC) TO THE
4515:        LABEL DESCRIBED BY #FLP'#. UPDATE THE RUNTIME STACK DISPLAY*) BEGIN
4516:       END (*GENLGJMP*);
4517:
4518:      BEGIN
4519:       IF SY = INTCONST THEN
4520:        BEGIN LLP := FSTLABP;
4521:         WHILE LLP <> FLABP DO (*DECIDE WHETHER LOCALLY DECLARED*)
4522:          WITH LLP' DO
4523:           IF LABVAL = IVAL THEN
4524:            BEGIN
4525:             IF DEFINED THEN
4526:              BEGIN
4527:               GOTODEF(LLP);
4528:               IF GLOBN < LABCONTR THEN ERROR(184);
4529:              END
4530:             ELSE
4531:              BEGIN
4532:               GOTONOTDEF(LLP);
4533:               IF GLOBN < LABCONTR THEN LABCONTR := GLOBN;
4534:              END;
4535:             GOTO 1
4536:            END
4537:           ELSE LLP := NEXTLAB;
4538:         WHILE LLP <> NIL DO (*DECIDE WHETHER GLOBALLY DECLARED*)
4539:          WITH LLP' DO
4540:           IF LABVAL = IVAL THEN
4541:            BEGIN
4542:             IF LCNT = 0 THEN
4543:              BEGIN LABCNT := LABCNT + 1;
4544:               LCNT := LABCNT
4545:              END;
4546:             IF FPROCP <> NIL THEN
4547:              WITH FPROCP' DO EXITLAB := EXITLAB + [LCNT];
4548:             GENLGJMP(LLP);
4549:             GOTO 1
4550:            END
4551:           ELSE LLP := NEXTLAB;
4552:         ERROR(167);
4553:   1:    INSYMBOL
4554:        END
```

```
4555:       ELSE ERROR(15);
4556:      END (*GOTOSTATEMENT*) ;
4557:
4558:      PROCEDURE COMPOUNDSTATEMENT;
4559:      BEGIN
4560:       REPEAT
4561:        STATEMENT(FSYS+[SEMICOLON,ENDSY]);
4562:        IF SY IN STATBEGSYS THEN ERROR(14);
4563:       UNTIL NOT (SY IN STATBEGSYS);
4564:       WHILE SY = SEMICOLON DO
4565:        BEGIN INSYMBOL;
4566:         REPEAT
4567:          STATEMENT(FSYS+[SEMICOLON,ENDSY]);
4568:          IF SY IN STATBEGSYS THEN ERROR(14);
4569:         UNTIL NOT (SY IN STATBEGSYS)
4570:        END;
4571:       IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
4572:      END (*COMPOUNDSTATEMENT*) ;
4573:
4574:      PROCEDURE IFSTATEMENT;
4575:       VAR LUSF1,LUSF2: USFREF; LATTRP: ATTRP;
4576:  (**)  (*.IFDECL.*)
4577:
4578:       PROCEDURE IFSTART;
4579:  (**) (*.IFSTART. THIS PROC IS CALLED WHEN AN #IF# IS DETECTED*) BEGIN
4580:       END (*IFSTART*);
4581:
4582:       PROCEDURE IFTEST(VAR FUSF1: USFREF);
4583:  (**) (*.IFTEST. PREPARE A FALSE JUMP ON #GATTRP'#. THE INSERTION
4584:        LOCATION IS DESCRIBED BY #FUSF1# *) BEGIN PREPFJMP(FUSF1);
4585:       END (*IFTEST*);
4586:
4587:       PROCEDURE IFELSE(VAR FUSF2: USFREF; FUSF1: USFREF);
4588:  (**) (*.IFELSE. THIS PROCEDURE IS CALLED ON DETECTION OF #ELSE#. PREPARE
4589:        AN UNCONDITIONAL JUMP AND RETURN INSERTION LOCATION IN #FUSF2#. SATISFY
4590:        THE REFERENCE DESCRIBED BY #FUSF1#.
4591:        *) BEGIN PREPJMP(FUSF2); SATISIC(FUSF1);
4592:       END (*IFELSE*);
4593:
4594:       PROCEDURE IFENDELSE(FUSF2: USFREF);
4595:  (**) (*.IFENDELSE. THIS PROCEDURE IS CALLED AT THE END OF AN #IF# STATEMENT
4596:         WITH AN #ELSE#. SATISFY THE REFENCE GIVEN BY #FUSF2#.
4597:        *) BEGIN SATISIC(FUSF2);
4598:       END (*IFENDELSE*);
4599:
4600:       PROCEDURE IFNOELSE(FUSF1: USFREF);
4601:  (**) (*.IFNOELSE. THIS PROCEDURE IS CALLED AT THE END OF AN #IF# STATEMENT
4602:        WITHOUT AN #ELSE#. SATISFY THE REFERENCE DESCRIBED BY #FUSF1#.
4603:        *) BEGIN SATISIC(FUSF1);
4604:       END (*IFNOELSE*);
4605:
4606:      BEGIN
4607:       IFSTART;
4608:       LATTRP := GATTRP;
4609:       EXPRESSION(FSYS+[THENSY]);
4610:       IF NOT COMPTYPES(GATTRP'.TYPTR,BOOLPTR) THEN ERROR(144);
4611:       IFTEST(LUSF1);
4612:       RESETGATTRP(LATTRP);
4613:       IF SY = THENSY THEN INSYMBOL ELSE ERROR(52);
4614:       STATEMENT(FSYS+[ELSESY]);
4615:       IF SY = ELSESY THEN
4616:        BEGIN
4617:         IFELSE(LUSF2,LUSF1);
4618:         INSYMBOL; STATEMENT(FSYS);
4619:         IFENDELSE(LUSF2);
4620:        END
```

```
4621:       ELSE
4622:        BEGIN
4623:         IFNOELSE(LUSF1);
4624:        END;
4625:      END (*IFSTATEMENT*) ;
4626:
4627:      PROCEDURE CASESTATEMENT;
4628:       LABEL 1;
4629:       TYPE CIP = 'CASEREC;
4630:          CASEREC = PACKED
4631:               RECORD NEXT: CIP;
4632:                CSLAB: INTEGER;
4633:                CSADDR: ADDRRANGE;
4634:               END;
4635:       VAR LSP,LSP1: STP; FSTPTR,LPT1,LPT2,LPT3: CIP; LVAL: VALU;
4636:         LSTART,LBND,LUSF: USFREF; LCSADDR,LERR: ADDRRANGE;
4637:         EXITLOOP: BOOLEAN; LUSFLIST: LOCOFREF;
4638:         LMIN,LMAX: INTEGER; LATTRP: ATTRP;
4639:  (**)   (*.CASEDECL. OTHER LOCAL DECLARATIONS*)
4640:
4641:         PROCEDURE CASESTART;
4642:  (**)   (*.CASESTART. THIS PROCEDURE IS CALLED AT THE BEGINNING OF THE #CASE#
4643:           STATEMENT *) BEGIN
4644:         END (*CASESTART*);
4645:
4646:         PROCEDURE PREPCHKBND(VAR FUSF: USFREF);
4647:  (**)   (*.PREPCHKBND. PREPARE THE CHECK OF #GATTRP'# AGAINST THE BOUNDS
4648:          WHICH WILL BE INSERTED BY #SATISBND#. RETURN THE INSERTION LOCATION
4649:          IN #FUSF#*) BEGIN
4650:         END (*PREPCHKBND*);
4651:
4652:         PROCEDURE CASETEST(VAR FSTART: USFREF);
4653:  (**)   (*.CASETEST. THIS PROCEDURE IS CALLED AFTER DETECTION OF THE #CASE#
4654:           EXPRESSION. PREPARE THE EXPRESSION DESCRIBED BY #GATTRP'# SO THAT
4655:           IT CAN BE USED IN THE SWITCH. PREPARE A JUMP TO THE SWITCH.
4656:           THE DESCRIPTION OF THE INSERTION LOCATION IS RETURNED IN
4657:           #FSTART# *) BEGIN
4658:         END (*CASETEST*);
4659:
4660:         PROCEDURE CASELABEL(VAR FCSADDR: ADDRRANGE);
4661:  (**)   (*.CASELABEL. THIS PROCEDURE IS CALLED AFTER DETECTION OF A LABEL
4662:           LIST. RETURN IN #FCSADDR# THE VALUE OF #IC# AT THIS POINT
4663:           (ONE WILL JUMP ONTO THAT ADDRESS) *) BEGIN FCSADDR := IC;
4664:         END (*CASELABEL*);
4665:
4666:         PROCEDURE CASEENDSTAT(VAR FUSFLIST: LOCOFREF);
4667:         VAR LUSF: USFREF;
4668:  (**)    (*.CASEENDSTAT. OTHER LOCAL DECLARATIONS*)
4669:         BEGIN
4670:  (**)    (*.CASEENDSTAT. THIS PROCEDURE IS CALLED AFTER A CASE SUBSTATEMENT.
4671:            PREPARE A JUMP AND PUT THE INSERTION LOCATION INTO THE LIST HEADED
4672:            BY #FUSFLIST# *) PREPJMP(LUSF); LINKOCC(FUSFLIST,LUSF);
4673:         END (*CASEENDSTAT*);
4674:
4675:         PROCEDURE GENERRHANDLING(VAR FERR: ADDRRANGE);
4676:  (**)   (*.GENERRHANDLING. GENERATE THE CODE TO BE EXECUTED WHEN THE #CASE#
4677:           EXPRESSION HAS A VALUE NOT GIVEN IN THE CONSTANT LIST. RETURN IN
4678:           #FERR# THE LOCATION OF THE BEGINNING OF THIS CODE*) BEGIN
4679:         END (*GENERRHANDLING*);
4680:
4681:         PROCEDURE SATISBND(FUSF: USFREF; FMIN,FMAX: INTEGER);
4682:  (**)   (*.SATISBND. INSERT THE BOUNDS #FMIN# AND FMAX# AT THE LOCATION
4683:           #FUSF# *) BEGIN
4684:         END (*SATISBND*);
4685:
4686:         PROCEDURE GENSWITCH(FCIP: CIP; FERR: ADDRRANGE);
```

```
4687:         VAR LVAL: INTEGER;
4688:  (**)    (*.GENSWITCH. OTHER LOCAL DECLARATIONS*)
4689:         BEGIN
4690:  (**)    (*.GENSWITCH. GENERATE INDEXED JUMP (FOR THE EXRESSION PREPARED IN
4691:            #CASETEST#) TO THE FOLLOWING SWITCH*)
4692:          LVAL := LMIN (*GLOBAL*);
4693:          (*SWITCH GENERATION*)
4694:          REPEAT
4695:           WITH FCIP' DO
4696:            BEGIN (*CASESTATEMENT*)
4697:             WHILE CSLAB > LVAL DO
4698:              BEGIN
4699:  (**)         (*.GENERRJMP. GENERATE THE JUMP TO THE ADDRESS #FERR# (GLOBAL)
4700:               *) GENJMP(FERR);
4701:              LVAL := LVAL + 1
4702:             END;
4703:  (**)       (*.GENNOERRJMP. GENERATE THE JUMP TO THE ADDRESS #CSADDR#
4704:              *) GENJMP(CSADDR);
4705:            LVAL := LVAL + 1; FCIP := NEXT
4706:            END
4707:          UNTIL FCIP = NIL
4708:         END (*GENSWITCH*);
4709:
4710:         PROCEDURE CASEEND(FUSFLIST: LOCOFREF);
4711:  (**)   (*.CASEEND. THIS PROCEDURE IS CALLED AT THE END OF THE #CASE#
4712:           STATEMENT. SATISFY ALL THE REFERENCES OF
4713:           #FUSFLIST# *) BEGIN SATISALLOCC(FUSFLIST);
4714:         END (*CASEEND*);
4715:
4716:      BEGIN (*CASESTATEMENT*)
4717:       CASESTART;
4718:       LATTRP := GATTRP;
4719:       EXPRESSION(FSYS+[OFSY,COMMA,COLON]);
4720:       LSP := GATTRP'.TYPTR; LUSFLIST := NIL;
4721:       IF LSP <> NIL THEN
4722:        IF (LSP'.FORM > SUBRANGE)OR COMPTYPES(LSP,REALPTR) THEN
4723:         BEGIN ERROR(144); LSP := NIL END;
4724:       PREPCHKBND(LBND);
4725:       CASETEST(LSTART);
4726:       RESETGATTRP(LATTRP);
4727:       IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
4728:       FSTPTR := NIL; LPT3 := NIL;
4729:       (*LOOP UNTIL SY <> SEMICOLON*)
4730:       REPEAT
4731:        IF NOT (SY IN [SEMICOLON,ENDSY]) THEN
4732:        BEGIN CASELABEL(LCSADDR);
4733:        (*LOOP UNTIL SY <> COMMA:*)
4734:        REPEAT CONSTANT(FSYS+[COMMA,COLON],LSP1,LVAL);
4735:         IF LSP1 <> NIL THEN
4736:          IF COMPTYPES(LSP,LSP1) THEN
4737:           BEGIN LPT1 := FSTPTR; LPT2 := NIL;
4738:            WHILE LPT1 <> NIL DO
4739:             WITH LPT1' DO
4740:              BEGIN
4741:               IF CSLAB <= LVAL.IVAL THEN
4742:                BEGIN IF CSLAB = LVAL.IVAL THEN ERROR(156);
4743:                 GOTO 1
4744:                END;
4745:               LPT2 := LPT1; LPT1 := NEXT
4746:              END;
4747:      1:     NEW(LPT3);
4748:            WITH LPT3' DO
4749:             BEGIN NEXT := LPT1; CSLAB := LVAL.IVAL;
4750:              CSADDR := LCSADDR;
4751:             END;
4752:            IF LPT2 = NIL THEN FSTPTR := LPT3
```

```
4753:            ELSE LPT2'.NEXT := LPT3
4754:           END
4755:          ELSE ERROR(147);
4756:        EXITLOOP := SY <> COMMA;
4757:         IF NOT EXITLOOP THEN INSYMBOL
4758:        UNTIL EXITLOOP;
4759:        IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
4760:        REPEAT STATEMENT(FSYS+[SEMICOLON]);
4761:         IF SY IN STATBEGSYS THEN ERROR(14);
4762:        UNTIL NOT (SY IN STATBEGSYS);
4763:        CASEENDSTAT(LUSFLIST);
4764:        END (*SY <> ENDSY*) ;
4765:        EXITLOOP := SY <> SEMICOLON;
4766:        IF NOT EXITLOOP THEN INSYMBOL;
4767:       UNTIL EXITLOOP;
4768:       GENERRHANDLING(LERR);
4769:       IF FSTPTR <> NIL THEN
4770:        BEGIN LMAX := FSTPTR'.CSLAB;
4771:         (*REVERSE POINTERS*)
4772:         LPT1 := FSTPTR; FSTPTR := NIL;
4773:         REPEAT LPT2 := LPT1'.NEXT; LPT1'.NEXT := FSTPTR;
4774:          FSTPTR := LPT1; LPT1 := LPT2
4775:         UNTIL LPT1 = NIL;
4776:         LMIN := FSTPTR'.CSLAB;
4777:         SATISBND(LBND,LMIN,LMAX);
4778:         SATISIC(LSTART);
4779:         GENSWITCH(FSTPTR,LERR);
4780:        END (*FSTPTR <> NIL*)
4781:       ELSE ERROR(6);
4782:       CASEEND(LUSFLIST);
4783:       IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
4784:      END (*CASESTATEMENT*) ;
4785:
4786:      PROCEDURE REPEATSTATEMENT;
4787:       VAR LADDR: ADDRRANGE; LATTRP: ATTRP;
4788:  (**) (*.REPEATDECL.*)
4789:
4790:       PROCEDURE REPEATSTART(VAR FADDR: ADDRRANGE);
4791:  (**) (*.REPEATSTART. THIS PROCEDURE IS CALLED AT THE BEGINNING OF A #REPEAT#
4792:         STATEMENT. RETURN IN #FADDR# THE ADDRESS TO BE JUMPED ON AT THE END
4793:         OF THE #REPEAT# LOOP *) BEGIN FADDR := IC;
4794:       END (*REPEATSTART*);
4795:
4796:       PROCEDURE REPEATUNTIL;
4797:  (**) (*.REPEATUNTIL. THIS PROCEDURE IS CALLED UPON DETECTION OF
4798:         #UNTIL# *) BEGIN
4799:       END (*REPEATUNTIL*);
4800:
4801:       PROCEDURE REPEATTEST(FADDR: ADDRRANGE);
4802:  (**) (*.REPEATTEST. THIS PROCEDURE IS CALLED AFTER DETECTION OF THE EXPRESSION
4803:         OF THE #REPEAT# STATEMENT. GENERATE A FALSE JUMP (ON #GATTRP'#) TO THE
4804:         ADDRESS GIVEN BY #FADDR# *) BEGIN GENFJMP(FADDR);
4805:       END (*REPEATTEST*);
4806:
4807:      BEGIN
4808:       REPEATSTART(LADDR);
4809:       (*LOOP UNTIL SY <> SEMICOLON:*)
4810:       REPEAT
4811:        STATEMENT(FSYS+[SEMICOLON,UNTILSY]);
4812:        IF SY IN STATBEGSYS THEN ERROR(14)
4813:       UNTIL NOT (SY IN STATBEGSYS);
4814:       WHILE SY = SEMICOLON DO
4815:        BEGIN INSYMBOL;
4816:         REPEAT
4817:          STATEMENT(FSYS+[SEMICOLON,UNTILSY])
4818:         UNTIL NOT (SY IN STATBEGSYS);
```

```
4819:        END;
4820:       IF SY = UNTILSY THEN
4821:        BEGIN INSYMBOL;
4822:         REPEATUNTIL;
4823:         LATTRP := GATTRP;
4824:         EXPRESSION(FSYS);
4825:         IF NOT COMPTYPES(GATTRP'.TYPTR,BOOLPTR) THEN ERROR(144);
4826:         REPEATTEST(LADDR);
4827:         RESETGATTRP(LATTRP);
4828:        END
4829:       ELSE ERROR(53);
4830:      END (*REPEATSTATEMENT*) ;
4831:
4832:      PROCEDURE WHILESTATEMENT;
4833:       VAR LADDR: ADDRRANGE; LUSF: USFREF; LATTRP: ATTRP;
4834:  (**)  (*.WHILEDECL.*)
4835:
4836:        PROCEDURE WHILESTART(VAR FADDR:ADDRRANGE);
4837:  (**)  (*.WHILESTART. THIS PROCEDURE IS CALLED UPON DETECTION OF #WHILE#.
4838:          SAVE IN #FADDR# THE LOCATION TO JUMP ON AFTER COMPLETION OF THE
4839:          LOOP *) BEGIN FADDR := IC;
4840:        END (*WHILESTART*);
4841:
4842:        PROCEDURE WHILETEST(VAR FUSF: USFREF);
4843:  (**)  (*.WHILETEST. THIS PROCEDRE IS CALLED AFTER THE EXPRESSION. PRE-
4844:          PARE A FALSE JUMP ON #GATTRP'#. THE DESCRIPTION OF THE INSERTION
4845:          LOCATION IS TO BE RETURNED IN #FUSF# *) BEGIN PREPFJMP(FUSF);
4846:        END (*WHILETEST*);
4847:
4848:        PROCEDURE WHILEEND(FADDR: ADDRRANGE; FUSF: USFREF);
4849:  (**)  (*.WHILEEND. THIS PROCEDURE IS CALLED AT THE END OF THE #WHILE# STATE-
4850:          MENT. GENERATE THE JUMP TO #FADDR# AND SATISFY THE REFERENCE DESCRIBED
4851:          BY #FUSF# *) BEGIN GENJMP(FADDR); SATISIC(FUSF);
4852:        END (*WHILEEND*);
4853:
4854:      BEGIN
4855:       WHILESTART(LADDR);
4856:       LATTRP := GATTRP;
4857:       EXPRESSION(FSYS+[DOSY]);
4858:       IF NOT COMPTYPES(GATTRP'.TYPTR,BOOLPTR) THEN ERROR(144);
4859:       WHILETEST(LUSF);
4860:       RESETGATTRP(LATTRP);
4861:       IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
4862:       STATEMENT(FSYS);
4863:       WHILEEND(LADDR,LUSF);
4864:      END (*WHILESTATEMENT*) ;
4865:
4866:      PROCEDURE FORSTATEMENT;
4867:       VAR LATTRP,FIRSTP,LIMITP,SAVLIMP,CONVARP: ATTRP; LSP: STP; LSY: SYMBOL;
4868:         LUSF: USFREF; LCP: CTP; ERRFLAG,NOTEST: BOOLEAN;
4869:         LMIN,LMAX: INTEGER; LIMADDR,LADDR: ADDRRANGE;
4870:  (**)   (*.FORDECL.*)
4871:
4872:         PROCEDURE FORSTART;
4873:  (**)   (*.FORSTART. THIS PROCEDURE IS CALLED UPON DETECTION OF THE
4874:           #FOR# *) BEGIN
4875:         END (*FORSTART*);
4876:
4877:         PROCEDURE CONTROLVARIABLE;
4878:         VAR LFORP: FORP; EXITLOOP: BOOLEAN;
4879:
4880:          PROCEDURE ATTRCONTRVAR(FCP: CTP);
4881:  (**)    (*.ATTRCONTRVAR. FILL THE ATTRIBUTE #GATTRP'# WITH THE INFOR-
4882:            MATION DESCRIBING THE CONTROL VARIABLE OF THE STATEMENT GIVEN BY
4883:            #FCP'# *) BEGIN
4884:          END (*ATTRCONTRVAR*);
```

```
4885:
4886:          PROCEDURE SETGATTRP;
4887:  (**)    (*.SETGATTRP. SET THE FIELDS OF #GATTRP'# TO AVOID UNDEFINED VALUES
4888:            THIS PROCEDRE IS CALLED AFTER AN ERROR *) BEGIN
4889:          END (*SETGATTRP*);
4890:
4891:         BEGIN ATTRNEW;
4892:          WITH LCP', GATTRP' DO
4893:           IF IDTYPE <> NIL THEN
4894:            IF (IDTYPE'.FORM>SUBRANGE) OR COMPTYPES(REALPTR,IDTYPE)
4895:             THEN BEGIN ERROR(143); SETGATTRP END
4896:            ELSE
4897:             BEGIN TYPTR := IDTYPE; INPCKDSTR := FALSE; KIND := VARBL;
4898:              EXPR := FALSE;
4899:              IF VKIND = DRCT THEN
4900:               BEGIN
4901:                IF NOT (LEVEL IN [1,LEVEL]) THEN ERROR(155);
4902:                ATTRCONTRVAR(LCP);
4903:               END
4904:              ELSE BEGIN ERROR(155); SETGATTRP END;
4905:              LFORP := FSTFORP'.NEXTFORP; EXITLOOP := LFORP = NIL;
4906:              WHILE NOT EXITLOOP DO
4907:               IF LFORP'.CONTROLID = LCP THEN
4908:                BEGIN ERROR(178); EXITLOOP := TRUE END
4909:               ELSE
4910:                BEGIN LFORP := LFORP'.NEXTFORP; EXITLOOP := LFORP = NIL
4911:                END;
4912:             END
4913:         END (*CONTROLVARIABLE*);
4914:
4915:         PROCEDURE TESTFOR(VAR FADDR: ADDRRANGE; VAR FUSF: USFREF);
4916:  (**)   (*.TESTFOR. GENERATE THE CODE AT THE BEGIN OF A #FOR# STATEMENT. THE
4917:           GLOBAL VARIABLES #CONVARP#,#FIRSTP#,#LIMITP# POINT TO THE ATTRIBUTES
4918:           DESCRIBING THE CONTROL VARIABLE, THE FIRST VALUE AND THE LIMIT.
4919:           THE GLOBAL VARIABLE #LSY# HAS THEN VALUE #TOSY# OR #DOWNTOSY#.
4920:           RETURN IN #FADDR# THEN ADDRESS WHERE TO JUMP AT THE END OF THE
4921:           STATEMENT TO REPEAT THE LOOP.
4922:           RETURN IN #FUSF# THE ADDRESS WHERE AN INSERTION HAS TO BE MADE
4923:           FOR THE FORWARD JUMP TO BE EXECUTED WHEN THE #FOR#
4924:           LOOP IS COMPLETED*) BEGIN
4925:         END (*TESTFOR*);
4926:
4927:         PROCEDURE ENDFOR(FADDR: ADDRRANGE; FUSF: USFREF);
4928:  (**)   (*.ENDFOR. GENERATE THE CODE AT THE END OF A #FOR# STATEMENT.
4929:           REFER TO #TESTFOR# FOR THE GLOBAL ATTRIBUTE POINTERS.
4930:           #FADDR#,#FUSF# ARE THE VALUES SET BY #TESTFOR#*) BEGIN
4931:         END (*ENDFOR*);
4932:
4933:      BEGIN (*FORSTATEMENT*)
4934:       FORPUSH;
4935:       LATTRP := GATTRP; ERRFLAG := FALSE;
4936:       FORSTART;
4937:       IF SY = IDENT THEN
4938:        BEGIN SEARCHID([VARS],LCP);
4939:         FSTFORP'.CONTROLID := LCP;
4940:         CONTROLVARIABLE; CONVARP := GATTRP;
4941:         NOTEST := FALSE;
4942:         IF LCP <> NIL THEN
4943:          IF LCP'.IDTYPE <> INTPTR THEN GETBOUNDS(LCP'.IDTYPE,LMIN,LMAX)
4944:          ELSE NOTEST := TRUE;
4945:         INSYMBOL
4946:        END
4947:       ELSE
4948:        BEGIN ERROR(2); SKIP(FSYS+[BECOMES,TOSY,DOWNTOSY,DOSY]) END;
4949:       IF SY = BECOMES THEN
4950:        BEGIN INSYMBOL; EXPRESSION(FSYS+[TOSY,DOWNTOSY,DOSY]);
```

```
4951:         FIRSTP := GATTRP;
4952:         IF GATTRP'.TYPTR <> NIL THEN
4953:           IF GATTRP'.TYPTR'.FORM > SUBRANGE THEN
4954:            BEGIN ERROR(144); ERRFLAG := TRUE END
4955:           ELSE
4956:            IF NOT COMPTYPES(CONVARP'.TYPTR,GATTRP'.TYPTR) THEN
4957:             BEGIN ERROR(145); ERRFLAG := TRUE END
4958:            ELSE
4959:             IF NOT NOTEST THEN
4960:              IF GATTRP'.KIND = CST THEN
4961:               BEGIN
4962:                IF (GATTRP'.CVAL.IVAL<LMIN) OR (GATTRP'.CVAL.IVAL>LMAX) THEN
4963:                 ERROR(302);
4964:               END
4965:              ELSE IF DEBUG THEN CHECKBNDS(LMIN,LMAX,2);
4966:        END
4967:       ELSE
4968:        BEGIN ERROR(51); SKIP(FSYS+[TOSY,DOWNTOSY,DOSY]); ERRFLAG := TRUE END;
4969:       LSY := SY;
4970:       IF SY IN [TOSY,DOWNTOSY] THEN
4971:        BEGIN
4972:         INSYMBOL; EXPRESSION(FSYS+[DOSY]);
4973:         LIMITP := GATTRP; GLOBN := GLOBN + 1;
4974:         IF GATTRP'.TYPTR <> NIL THEN
4975:         IF GATTRP'.TYPTR'.FORM > SUBRANGE THEN
4976:          BEGIN ERROR(144); ERRFLAG := TRUE END
4977:          ELSE
4978:           IF NOT COMPTYPES(LIMITP'.TYPTR,CONVARP'.TYPTR) THEN
4979:            BEGIN ERROR(145); ERRFLAG := TRUE END
4980:           ELSE
4981:            IF NOT NOTEST THEN
4982:             IF GATTRP'.KIND = CST THEN
4983:              BEGIN
4984:               IF (GATTRP'.CVAL.IVAL<LMIN) OR (GATTRP'.CVAL.IVAL>LMAX) THEN
4985:                ERROR(303);
4986:              END
4987:             ELSE IF DEBUG THEN CHECKBNDS(LMIN,LMAX,2);
4988:        END
4989:       ELSE BEGIN ERROR(55); SKIP(FSYS+[DOSY]); ERRFLAG := TRUE END;
4990:       IF SY = DOSY THEN INSYMBOL ELSE BEGIN ERRFLAG := TRUE; ERROR(54) END;
4991:       IF NOT ERRFLAG THEN TESTFOR(LADDR,LUSF);
4992:       STATEMENT(FSYS);
4993:       SETLCMAX;
4994:       IF NOT ERRFLAG THEN ENDFOR(LADDR,LUSF);
4995:       RESETGATTRP(LATTRP);
4996:       GLOBN := GLOBN - 1; CHECKLABELS;
4997:       FORPOP;
4998:      END (*FORSTATEMENT*) ;
4999:
5000:      PROCEDURE WITHSTATEMENT;
5001:       VAR LCP: CTP; OLDTOP: DISPRANGE; P,LATTRP: ATTRP;
5002:         EXITLOOP: BOOLEAN; LGLOBN: INTEGER;
5003:  (**)   (*.WITHDECL. OTHER LOCAL DECLARATIONS*)
5004:
5005:       PROCEDURE CONVGATTRP;
5006:  (**) (*.CONVGATTRP. CONVERT THE ATTRIBUTE #GATTRP'# INTO A FORM SUITED FOR
5007:         AN ARGUMENT OF A WITH STATEMENT*) BEGIN
5008:       END (*CONVGATTRP*);
5009:
5010:       PROCEDURE ENDWITH;
5011:  (**) (*.ENDWITH. THIS PROCEDURE IS CALLED AT THE END OF THE #WITH#
5012:         STATEMENT *) BEGIN
5013:       END (*ENDWITH*);
5014:
5015:      BEGIN OLDTOP := TOP; LATTRP := GATTRP; LGLOBN := GLOBN;
5016:       (*LOOP UNTIL SY <> COMMA:*)
```

```
5017:       REPEAT
5018:        IF SY = IDENT THEN
5019:         BEGIN SEARCHID([VARS,FIELD],LCP); INSYMBOL END
5020:        ELSE BEGIN ERROR(2); LCP := UVARPTR END;
5021:        SELECTOR(FSYS+[COMMA,DOSY],LCP);
5022:        IF GATTRP'.TYPTR <> NIL THEN
5023:         IF GATTRP'.TYPTR'.FORM = RECORDS THEN
5024:          IF TOP < DISPLIMIT THEN
5025:           BEGIN TOP := TOP + 1;
5026:            WITH DISPLAY[TOP],GATTRP' DO
5027:             BEGIN FNAME := TYPTR'.FIELDS;
5028:              OCCUR := REC;
5029:              GLOBN := GLOBN + 1;
5030:              CONVGATTRP;
5031:              WITHATTRP := GATTRP;
5032:             END
5033:           END
5034:          ELSE ERROR(250)
5035:         ELSE ERROR(140);
5036:        EXITLOOP := SY <> COMMA;
5037:        IF NOT EXITLOOP THEN INSYMBOL
5038:       UNTIL EXITLOOP;
5039:       SETLCMAX;
5040:       IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
5041:       STATEMENT(FSYS);
5042:       ENDWITH;
5043:       RESETGATTRP(LATTRP); GLOBN := LGLOBN;
5044:       CHECKLABELS;
5045:       TOP := OLDTOP;
5046:      END (*WITHSTATEMENT*);
5047:
5048:      PROCEDURE DEFGLOBLAB(FLP: LBP);
5049:   (**) (*.DEFGLOBLAB. DEFINE THE GLOBAL LABEL #FLP#*) BEGIN
5050:      END (*DEFGLOBLAB*);
5051:
5052:      PROCEDURE STATLABEL;
5053:   (**) (*.STATLABEL. THIS PROCEDURE IS CALLED UPON DETECTION OF A LABEL*)BEGIN
5054:      END (*STATLABEL*);
5055:
5056:     BEGIN (*STATEMENT*)
5057:      IF SY = INTCONST THEN (*LABEL*)
5058:       BEGIN
5059:        LLP := FSTLABP;
5060:        WHILE LLP <> FLABP DO
5061:         WITH LLP' DO
5062:          IF LABVAL = IVAL THEN
5063:           BEGIN
5064:            IF DEFINED THEN ERROR(165)
5065:            ELSE
5066:             BEGIN LOCP := FSTOCC;
5067:              IF LOCP <> NIL THEN
5068:               IF GLOBN > LABCONTR THEN ERROR(184);
5069:              WHILE LOCP <> NIL DO
5070:               WITH LOCP' DO
5071:                BEGIN SATISIC(LOC);
5072:                 LOCP := NXTREF;
5073:                END;
5074:              IF LCNT <> 0 THEN DEFGLOBLAB(LLP);
5075:              DEFINED := TRUE; LABADDR := IC; LABCONTR := GLOBN;
5076:             END;
5077:            GOTO 1
5078:           END
5079:          ELSE LLP := NEXTLAB;
5080:        ERROR(167);
5081:      1: STATLABEL;
5082:        INSYMBOL;
```

```
5083:        IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
5084:       END;
5085:      IF NOT (SY IN FSYS+[IDENT]) THEN
5086:       BEGIN ERROR(6); SKIP(FSYS) END;
5087:      IF SY IN STATBEGSYS+[IDENT] THEN
5088:       BEGIN
5089:        CASE SY OF
5090:         IDENT:    BEGIN SEARCHID([VARS,FIELD,FUNC,PROC],LCP); INSYMBOL;
5091:                    IF LCP'.KLASS = PROC THEN CALL(FSYS,LCP)
5092:                    ELSE ASSIGNMENT(LCP)
5093:                   END;
5094:         BEGINSY:  BEGIN INSYMBOL; COMPOUNDSTATEMENT END;
5095:         GOTOSY:   BEGIN INSYMBOL; GOTOSTATEMENT END;
5096:         IFSY:     BEGIN INSYMBOL; IFSTATEMENT END;
5097:         CASESY:   BEGIN INSYMBOL; CASESTATEMENT END;
5098:         WHILESY:  BEGIN INSYMBOL; WHILESTATEMENT END;
5099:         REPEATSY: BEGIN INSYMBOL; REPEATSTATEMENT END;
5100:         FORSY:    BEGIN INSYMBOL; FORSTATEMENT END;
5101:         WITHSY:   BEGIN INSYMBOL; WITHSTATEMENT END
5102:        END;
5103:        IF NOT (SY IN FSYS) THEN
5104:         BEGIN ERROR(6); SKIP(FSYS) END
5105:       END;
5106:     END (*STATEMENT*) ;
5107:
5108:    BEGIN (*BODY*)
5109:     DP := FALSE;
5110:     BODYINITIALIZE;
5111:     IF LEVEL = 1 THEN
5112:      BEGIN MAINBODYINIT; OPENCODEGEN;
5113:       PREPOVFLWTEST(LUSF);
5114:       OPENFILES(DISPLAY[1].FNAME);
5115:       EXFILP := FEXFILP; FIRST := TRUE;
5116:       WHILE EXFILP <> NIL DO
5117:        WITH EXFILP' DO
5118:         BEGIN
5119:          IF NOT DECLARED AND (FILENAME <> #INPUT   #) AND
5120:            (FILENAME <> #OUTPUT  #) THEN
5121:           BEGIN
5122:            IF FIRST THEN
5123:             BEGIN ERROR(172); LCHCNT := CHCNT; ENDOFLINE;
5124:              FIRST := FALSE
5125:             END;
5126:            WRITELN(# FILE-ID #,FILENAME)
5127:           END;
5128:          EXFILP := NXTP
5129:         END;
5130:        IF NOT FIRST THEN
5131:         BEGIN CHCNT := LCHCNT;
5132:          FOR LCHCNT := 1 TO CHCNT DO LINE[LCHCNT] := # #
5133:         END
5134:     END
5135:    ELSE
5136:     BEGIN
5137:      PFBODYINIT(FPROCP); OPENCODEGEN;
5138:      PREPOVFLWTEST(LUSF);
5139:      COPYPARAMETERS;
5140:      OPENFILES(DISPLAY[TOP].FNAME);
5141:     END;
5142:      REPEAT STATEMENT(FSYS+[SEMICOLON,ENDSY]);
5143:       IF SY IN STATBEGSYS THEN ERROR(14)
5144:      UNTIL NOT (SY IN STATBEGSYS);
5145:      WHILE SY = SEMICOLON DO
5146:       BEGIN INSYMBOL;
5147:        REPEAT STATEMENT(FSYS+[SEMICOLON,ENDSY]);
5148:         IF SY IN STATBEGSYS THEN ERROR(14)
```

```
5149:         UNTIL NOT (SY IN STATBEGSYS)
5150:        END;
5151:       IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
5152:       FIRST := TRUE;
5153:       WHILE FSTLABP <> FLABP DO
5154:        WITH FSTLABP' DO
5155:         BEGIN
5156:          IF DEFINED THEN
5157:           BEGIN IF LCNT <> 0 THEN LGJMPLAB(FSTLABP) END
5158:          ELSE
5159:           BEGIN
5160:            IF FIRST THEN
5161:             BEGIN ERROR(168); LCHCNT := CHCNT; ENDOFLINE; FIRST := FALSE
5162:             END;
5163:            WRITELN(# LABEL #,LABVAL)
5164:           END;
5165:          FSTLABP := NEXTLAB
5166:         END;
5167:       IF NOT FIRST THEN
5168:        BEGIN CHCNT := LCHCNT;
5169:         FOR LCHCNT := 1 TO CHCNT DO LINE[LCHCNT] := # #
5170:        END;
5171:       CLOSEFILES;
5172:       RETURN(FPROCP);
5173:       SATISOVFLW(LUSF);
5174:       CLOSECODEGEN;
5175:     END (*BODY*);
5176:
5177:   BEGIN (*BLOCK*)
5178:    FLABP := FSTLABP; LFORWCNT := 0;
5179:    REPEAT
5180:     IF SY = LABELSY THEN
5181:      BEGIN INSYMBOL; LABELDECLARATION END;
5182:     IF SY = CONSTSY THEN
5183:      BEGIN INSYMBOL; CONSTDECLARATION END;
5184:     IF SY = TYPESY THEN
5185:      BEGIN INSYMBOL; TYPEDECLARATION END;
5186:     IF SY = VARSY THEN
5187:      BEGIN INSYMBOL; VARDECLARATION END;
5188:     WHILE SY IN [PROCSY,FUNCTSY] DO
5189:      BEGIN LSY := SY; INSYMBOL; PROCDECLARATION(LSY) END;
5190:     IF SY <> BEGINSY THEN
5191:      BEGIN ERROR(18); SKIP(FSYS) END
5192:    UNTIL SY IN STATBEGSYS;
5193:    IF LFORWCNT > 0 THEN
5194:     BEGIN LCHCNT := CHCNT; ENDOFLINE;
5195:      CHECKFORW(DISPLAY[LEVEL].FNAME);
5196:      CHCNT := LCHCNT;
5197:      FOR LCHCNT := 1 TO CHCNT DO LINE[LCHCNT] := # #;
5198:     END;
5199:    IF P THEN PRINTTABLES(LEVEL=1);
5200:    IF SY = BEGINSY THEN INSYMBOL ELSE ERROR(17);
5201:    REPEAT BODY(FSYS+[CASESY]);
5202:     IF SY <> FSY THEN
5203:      BEGIN ERROR(6); SKIP(FSYS) END
5204:    UNTIL (SY = FSY)OR (SY IN BLOCKBEGSYS);
5205:   END (*BLOCK*) ;
5206:
5207:
5208:   PROCEDURE PROGRAMME(FSYS: SETOFSYS);
5209:    LABEL 1;
5210:    VAR EXFILP: EXTFILEP; LCP: CTP;
5211:
5212:    PROCEDURE STARTCODEGEN;
5213:   (**) (*.STARTCODEGEN. START THE GENERATION OF CODE*) BEGIN
5214:    END (*STARTCODEGEN*);
```

```
5215:
5216:    PROCEDURE ENDCODEGEN;
5217:  (**) (*.ENDCODEGEN. END THE GENERATION OF CODE*) BEGIN
5218:    END (*ENDCODEGEN*);
5219:
5220:    PROCEDURE INPUTADDRESS(FCP: CTP);
5221:  (**) (*.INPUTADDRESS. GIVE THE ADDRESS OF THE INPUT FILE TO #FCP'.VADDR#.
5222:          CORRECT #LC# ACCORDINGLY*) BEGIN
5223:    END (*INPUTADDRESS*);
5224:
5225:    PROCEDURE OUTPUTADDRESS(FCP: CTP);
5226:  (**) (*.OUTPUTADDRESS. GIVE THE ADDRESS OF THE OUTPUT FILE TO #FCP'.VADDR#.
5227:          CORRECT #LC# ACCORDINGLY*) BEGIN
5228:    END (*OUTPUTADDRESS*);
5229:
5230:   BEGIN PROGNAME := #P.MAIN  #; FEXFILP := NIL; EXTFILS := 0;
5231:    (*DECLARE OUTPUT*)
5232:    NEW(LCP,VARS);
5233:    WITH LCP' DO
5234:     BEGIN NAME := #OUTPUT  #; IDTYPE := TEXTPTR;
5235:      KLASS := VARS; VKIND := DRCT; NEXT := NIL;
5236:      VLEV := 1; OUTPUTADDRESS(LCP)
5237:     END;
5238:    ENTERID(LCP);
5239:    STARTCODEGEN;
5240:    IF SY = PROGRAMSY THEN
5241:     BEGIN INSYMBOL;
5242:      IF SY = IDENT THEN
5243:       BEGIN PROGNAME := ID; INSYMBOL;
5244:        IF NOT (SY IN [SEMICOLON,LPARENT]) THEN
5245:         BEGIN ERROR(7); SKIP(FSYS+[SEMICOLON,LPARENT]) END;
5246:        IF SY = LPARENT THEN
5247:         BEGIN
5248:          REPEAT INSYMBOL;
5249:           IF SY = IDENT THEN
5250:            BEGIN
5251:             IF ID = #INPUT   # THEN
5252:              BEGIN NEW(INPUTPTR,VARS);
5253:               WITH INPUTPTR' DO
5254:                BEGIN NAME := #INPUT   #; IDTYPE := TEXTPTR;
5255:                 KLASS := VARS; VKIND := DRCT; NEXT := NIL;
5256:                 VLEV := 1; INPUTADDRESS(INPUTPTR);
5257:                END;
5258:               ENTERID(INPUTPTR);
5259:              END
5260:             ELSE
5261:              IF ID = #OUTPUT  # THEN OUTPUTPTR := LCP;
5262:             EXTFILS := EXTFILS + 1;
5263:             EXFILP := FEXFILP;
5264:             WHILE EXFILP <> NIL DO
5265:              WITH EXFILP' DO
5266:               BEGIN
5267:                IF FILENAME = ID THEN
5268:                 BEGIN ERROR(101); GOTO 1 END;
5269:                EXFILP := NXTP
5270:               END;
5271:   1:        NEW(EXFILP);
5272:             WITH EXFILP' DO
5273:              BEGIN FILENAME := ID; NXTP := FEXFILP;
5274:               DECLARED := FALSE;
5275:               INSYMBOL;
5276:              END;
5277:             FEXFILP := EXFILP
5278:            END
5279:           ELSE ERROR(2);
5280:           IF NOT (SY IN [COMMA,RPARENT]) THEN
```

```
5281:            BEGIN ERROR(6); SKIP(FSYS+[COMMA,RPARENT]) END
5282:          UNTIL SY <> COMMA;
5283:          IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
5284:         END;
5285:        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
5286:       END
5287:      ELSE BEGIN ERROR(2); SKIP(FSYS) END
5288:     END
5289:    ELSE BEGIN ERROR(3); SKIP(FSYS) END;
5290:    IF OUTPUTPTR = NIL THEN
5291:     BEGIN ERROR(176); OUTPUTPTR := LCP END;
5292:    REPEAT BLOCK(FSYS,PERIOD,NIL)
5293:    UNTIL SY = PERIOD;
5294:    ENDCODEGEN;
5295:   END (*PROGRAMME*) ;
5296:
5297:   PROCEDURE STDNAMES;
5298:   BEGIN
5299:    NA[ 1] := #FALSE   #; NA[ 2] := #TRUE    #; NA[ 3] := #OUTPUT  #;
5300:    NA[ 4] := #INPUT   #; NA[ 5] := #GET     #; NA[ 6] := #PUT     #;
5301:    NA[ 7] := #RESET   #; NA[ 8] := #REWRITE #; NA[ 9] := #PAGE    #;
5302:    NA[10] := #READ    #; NA[11] := #READLN  #; NA[12] := #WRITE   #;
5303:    NA[13] := #WRITELN #; NA[14] := #PACK    #; NA[15] := #UNPACK  #;
5304:    NA[16] := #NEW     #; NA[17] := #MARK    #; NA[18] := #RELEASE #;
5305:    NA[19] := #EOF     #; NA[20] := #EOLN    #; NA[21] := #ODD     #;
5306:    NA[22] := #ROUND   #; NA[23] := #TRUNC   #; NA[24] := #ABS     #;
5307:    NA[25] := #SQR     #; NA[26] := #ORD     #; NA[27] := #CHR     #;
5308:    NA[28] := #PRED    #; NA[29] := #SUCC    #; NA[30] := #SIN     #;
5309:    NA[31] := #COS     #; NA[32] := #EXP     #; NA[33] := #SQRT    #;
5310:    NA[34] := #LN      #; NA[35] := #ARCTAN  #;
5311:   END (*STDNAMES*) ;
5312:
5313:   PROCEDURE STDTYPENTRIES;
5314:   BEGIN                                                    (*TYPE*)
5315:                                                            (******)
5316:
5317:    NEW(INTPTR,SCALAR,STANDARD);                             (*INTEGER*)
5318:    WITH INTPTR' DO
5319:     BEGIN FORM := SCALAR; SCALKIND := STANDARD; FTYPE := FALSE;
5320:  (**) (*.STDTYPENTRIES. FILL IN #SIZE#*)
5321:     END;
5322:    NEW(REALPTR,SCALAR,STANDARD);                            (*REAL*)
5323:    WITH REALPTR' DO
5324:     BEGIN FORM := SCALAR; SCALKIND := STANDARD; FTYPE := FALSE;
5325:  (**) (*.STDTYPENTRIES. FILL IN #SIZE#*)
5326:     END;
5327:    NEW(CHARPTR,SCALAR,STANDARD);                            (*CHAR*)
5328:    WITH CHARPTR' DO
5329:     BEGIN FORM := SCALAR; SCALKIND := STANDARD; FTYPE := FALSE;
5330:  (**) (*.STDTYPENTRIES. FILL IN #SIZE#*)
5331:     END;
5332:    NEW(BOOLPTR,SCALAR,DECLARED);                            (*BOOLEAN*)
5333:    WITH BOOLPTR' DO
5334:     BEGIN FORM := SCALAR; SCALKIND := DECLARED; FTYPE := FALSE;
5335:  (**) (*.STDTYPENTRIES. FILL IN #SIZE#*)
5336:     END;
5337:    NEW(NILPTR,POINTER);                                     (*NIL*)
5338:    WITH NILPTR' DO
5339:     BEGIN ELTYPE := NIL; FORM := POINTER; FTYPE := FALSE;
5340:  (**) (*.STDTYPENTRIES. FILL IN #SIZE#*)
5341:     END;
5342:    NEW(TEXTPTR,FILES);                                      (*TEXT*)
5343:    WITH TEXTPTR' DO
5344:     BEGIN FILTYPE := CHARPTR; PCKDFIL := TRUE; FORM := FILES;
5345:      (*NOTE THAT IN THIS IMPLEMENTATION THE TEXT FILE IS ASSUMED TO BE PACKED*)
5346:      TEXTFILE := TRUE; FTYPE := TRUE;
```

```
5347:  (**) (*.STDTYPENTRIES. FILL IN #SIZE#*)
5348:     END;
5349:   END (*STDTYPENTRIES*);
5350:
5351:   PROCEDURE STDNAMENTRIES;
5352:    VAR CP,CP1: CTP; I,LCNT: INTEGER;
5353:   BEGIN                                                    (*NAME:*)
5354:                                                            (*******)
5355:
5356:    NEW(CP,TYPES);                                         (*INTEGER*)
5357:    WITH CP' DO
5358:     BEGIN NAME := #INTEGER #; IDTYPE := INTPTR; KLASS := TYPES END;
5359:    ENTERID(CP);
5360:    NEW(CP,TYPES);                                         (*REAL*)
5361:    WITH CP' DO
5362:     BEGIN NAME := #REAL    #; IDTYPE := REALPTR; KLASS := TYPES END;
5363:    ENTERID(CP);
5364:    NEW(CP,TYPES);                                         (*CHAR*)
5365:    WITH CP' DO
5366:     BEGIN NAME := #CHAR    #; IDTYPE := CHARPTR; KLASS := TYPES END;
5367:    ENTERID(CP);
5368:    NEW(CP,TYPES);                                         (*BOOLEAN*)
5369:    WITH CP' DO
5370:     BEGIN NAME := #BOOLEAN #; IDTYPE := BOOLPTR; KLASS := TYPES END;
5371:    ENTERID(CP);
5372:    NEW(CP,TYPES);                                         (*TEXT*)
5373:    WITH CP' DO
5374:     BEGIN NAME := #TEXT    #; IDTYPE := TEXTPTR; KLASS := TYPES END;
5375:    ENTERID(CP);
5376:    NEW(CP,KONST);                                         (*NIL*)
5377:    WITH CP' DO
5378:     BEGIN NAME := #NIL     #; IDTYPE := NILPTR; KLASS := KONST;
5379:      NEXT := NIL; VALUES.IVAL := NILVAL;
5380:     END;
5381:    ENTERID(CP);
5382:    NEW(CP,KONST);                                         (*MAXINT*)
5383:    WITH CP' DO
5384:     BEGIN NAME := #MAXINT  #; IDTYPE := INTPTR; KLASS := KONST;
5385:      NEXT := NIL; VALUES.IVAL := MXINT;
5386:     END;
5387:    ENTERID(CP);
5388:    CP1 := NIL; LCNT := 0;
5389:    FOR I := 1 TO 2 DO
5390:     BEGIN NEW(CP,KONST);                                 (*FALSE,TRUE*)
5391:      WITH CP' DO
5392:       BEGIN NAME := NA[LCNT+I]; IDTYPE := BOOLPTR; KLASS := KONST;
5393:        NEXT := CP1; VALUES.IVAL := I - 1
5394:       END;
5395:      ENTERID(CP); CP1 := CP
5396:     END;
5397:    LCNT := LCNT + 4;
5398:    BOOLPTR'.FCONST := CP;
5399:    FOR I := 1 TO NRSTDPROC DO
5400:     BEGIN NEW(CP,PROC,STANDARD);                 (*GET,PUT,RESET*)
5401:      WITH CP' DO                                 (*REWRITE,PAGE*)
5402:       BEGIN NAME := NA[LCNT+I]; IDTYPE := NIL;   (*READ,READLN*)
5403:        NEXT := NIL; KEY := I;                    (*WRITE,WRITELN*)
5404:        KLASS := PROC; PFDECKIND := STANDARD      (*PACK,UNPACK,NEW*)
5405:       END;                                       (*MARK,RELEASE*)
5406:      ENTERID(CP)
5407:     END;
5408:    LCNT := LCNT + NRSTDPROC;
5409:    FOR I := 1 TO NRSTDFUNC DO                   (*EOF,EOLN,ODD*)
5410:     BEGIN NEW(CP,FUNC,STANDARD);                (*ROUND,TRUNC*)
5411:      WITH CP' DO                                 (*ABS,SQR,ORD*)
5412:       BEGIN NAME := NA[LCNT+I]; IDTYPE := NIL;   (*CHR,PRED,SUCC*)
```

```
5413:        KLASS := FUNC; PFDECKIND := STANDARD;
5414:         NEXT := NIL; KEY := I;
5415:      END;
5416:      ENTERID(CP)
5417:     END;
5418:    LCNT := LCNT + NRSTDFUNC;
5419:    FOR I := 1 TO NREXFUNC DO                        (*SIN,COS,EXP*)
5420:     BEGIN NEW(CP,FUNC,STANDARD);                    (*SQRT,LN,ARCTAN*)
5421:      WITH CP' DO
5422:       BEGIN NAME := NA[LCNT+I]; IDTYPE := REALPTR;
5423:        KLASS := FUNC; PFDECKIND := STANDARD;
5424:        NEXT := NIL; KEY := NRSTDFUNC + I
5425:       END;
5426:      ENTERID(CP)
5427:     END;
5428:   END (*STDNAMENTRIES*);
5429:
5430:   PROCEDURE ENTERUNDECL;
5431:   BEGIN
5432:    NEW(UTYPPTR,TYPES);
5433:    WITH UTYPPTR' DO
5434:     BEGIN NAME := #        #; IDTYPE := NIL; KLASS := TYPES END;
5435:    NEW(UCSTPTR,KONST);
5436:    WITH UCSTPTR' DO
5437:     BEGIN NAME := #        #; IDTYPE := NIL; NEXT := NIL;
5438:      VALUES.IVAL := 0; KLASS := KONST;
5439:     END;
5440:    NEW(UVARPTR,VARS);
5441:    WITH UVARPTR' DO
5442:     BEGIN NAME := #        #; IDTYPE := NIL; VKIND := DRCT;
5443:      NEXT := NIL; VLEV := 0; VADDR := 0; KLASS := VARS
5444:     END;
5445:    NEW(UFLDPTR,FIELD);
5446:    WITH UFLDPTR' DO
5447:     BEGIN NAME := #        #; IDTYPE := NIL; NEXT := NIL; PCKDFLD := FALSE;
5448:      FLDADDR := 0; KLASS := FIELD
5449:     END;
5450:    NEW(UPRCPTR,PROC,DECLARED,ACTUAL);
5451:    WITH UPRCPTR' DO
5452:     BEGIN NAME := #        #; IDTYPE := NIL;
5453:      KLASS := PROC; PFDECKIND := DECLARED; PFKIND := ACTUAL;
5454:      NEXT := NIL; PFDECL := DEF; PFLEV := 0; PFADDR := 0
5455:     END;
5456:    NEW(UFCTPTR,FUNC,DECLARED,ACTUAL);
5457:    WITH UFCTPTR' DO
5458:     BEGIN NAME := #        #; IDTYPE := NIL; NEXT := NIL;
5459:      KLASS := FUNC; PFDECKIND := DECLARED; PFKIND := ACTUAL;
5460:      PFDECL := DEF; PFLEV := 0; PFADDR := 0
5461:     END
5462:   END (*ENTERUNDECL*) ;
5463:
5464:   PROCEDURE INITSCALARS;
5465:   VAR I: INTEGER;
5466:   BEGIN
5467:    KK := ALFALENG;
5468:    CH := # #; EOL := EOLN(INPUT); CHCNT := 0;
5469:    LC := LCPROGINIT; IC := ICSTART;
5470:    LABCNT := 0; EXTFILS := 0; FILECNT := 0; PCNT := 0;
5471:    DP := TRUE; PRTERR := TRUE;
5472:    DEBUG := TRUE; LISTON := TRUE; PMD := TRUE;
5473:    P := TRUE; (*FOR TEST PHASE ONLY*)
5474:    TERMINATED := FALSE;
5475:    INPUTPTR := NIL; OUTPUTPTR := NIL;
5476:    FWPTR := NIL; FSTLABP := NIL; FEXFILP := NIL;
5477:    ERRINX := 0; ERRORS := FALSE;
5478:    LINECOUNT := 0;
```

```
5479:    FOR I := 1 TO STRGFRL DO BLANKSTRING[I] := # #;
5480:    FOR I := 1 TO MAXCHCNT DO SKIPLINE[I] := # #;
5481:    SKIPPING := FALSE; SKIPINLINE := FALSE;
5482:    LCORIC := LC;
5483:    LASTCHCNT := 1;
5484:  (**) (*.INITSCALARS. INITIALIZE OTHER SCALARS*)
5485:   END (*INITSCALARS*);
5486:
5487:   PROCEDURE INITSETS;
5488:   BEGIN
5489:    CONSTBEGSYS := [ADDOP,INTCONST,REALCONST,CHARCONST,STRINGCONST,IDENT];
5490:    SIMPTYPEBEGSYS := [LPARENT]+CONSTBEGSYS;
5491:    TYPEBEGSYS := [ARROW,PACKEDSY,ARRAYSY,RECORDSY,SETSY,
5492:           FILESY]+SIMPTYPEBEGSYS;
5493:    TYPEDELS := [ARRAYSY,RECORDSY,SETSY,FILESY];
5494:    BLOCKBEGSYS := [LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,FUNCTSY,
5495:            BEGINSY];
5496:    SELECTSYS := [ARROW,PERIOD,LBRACK];
5497:    FACBEGSYS := [INTCONST,REALCONST,CHARCONST,STRINGCONST,IDENT,LPARENT,
5498:           LBRACK,NOTSY];
5499:    STATBEGSYS := [BEGINSY,GOTOSY,IFSY,WHILESY,REPEATSY,FORSY,WITHSY,
5500:            CASESY];
5501:  (**) (*.INITSETS. INITIALIZE OTHER SETS*)
5502:   END (*INITSETS*) ;
5503:
5504:   PROCEDURE INITTABLES;
5505:
5506:    PROCEDURE RESERVEDWORDS;
5507:    BEGIN
5508:     RW[ 1] := #IF      #; RW[ 2] := #DO      #; RW[ 3] := #OF      #;
5509:     RW[ 4] := #TO      #; RW[ 5] := #IN      #; RW[ 6] := #OR      #;
5510:     RW[ 7] := #END     #; RW[ 8] := #FOR     #; RW[ 9] := #VAR     #;
5511:     RW[10] := #DIV     #; RW[11] := #MOD     #; RW[12] := #SET     #;
5512:     RW[13] := #AND     #; RW[14] := #NOT     #; RW[15] := #THEN    #;
5513:     RW[16] := #ELSE    #; RW[17] := #WITH    #; RW[18] := #GOTO    #;
5514:     RW[19] := #CASE    #; RW[20] := #TYPE    #; RW[21] := #FILE    #;
5515:     RW[22] := #BEGIN   #; RW[23] := #UNTIL   #; RW[24] := #WHILE   #;
5516:     RW[25] := #ARRAY   #; RW[26] := #CONST   #; RW[27] := #LABEL   #;
5517:     RW[28] := #REPEAT  #; RW[29] := #RECORD  #; RW[30] := #DOWNTO  #;
5518:     RW[31] := #PACKED  #; RW[32] := #PROGRAM #; RW[33] := #FUNCTION#;
5519:     RW[34] := #PROCEDUR#;
5520:     LRW[0] :=  0; LRW[1] :=  0; LRW[2] :=  6; LRW[3] := 14; LRW[4] := 21;
5521:     LRW[5] := 27; LRW[6] := 31; LRW[7] := 32; LRW[8] := 34;
5522:    END (*RESWORDS*) ;
5523:
5524:    PROCEDURE SYMBOLS;
5525:    BEGIN
5526:     RSY[1] := IFSY; RSY[2] := DOSY; RSY[3] := OFSY; RSY[4] := TOSY;
5527:     RSY[5] := RELOP; RSY[6] := ADDOP; RSY[7] := ENDSY; RSY[8] := FORSY;
5528:     RSY[9] := VARSY; RSY[10] := MULOP;RSY[11] := MULOP; RSY[12] := SETSY;
5529:     RSY[13] := MULOP; RSY[14] := NOTSY; RSY[15] := THENSY;
5530:     RSY[16] := ELSESY; RSY[17] := WITHSY; RSY[18] := GOTOSY;
5531:     RSY[19] := CASESY; RSY[20] := TYPESY; RSY[21] := FILESY;
5532:     RSY[22] := BEGINSY; RSY[23] := UNTILSY; RSY[24] := WHILESY;
5533:     RSY[25] := ARRAYSY; RSY[26] := CONSTSY; RSY[27] := LABELSY;
5534:     RSY[28] := REPEATSY; RSY[29] := RECORDSY; RSY[30] := DOWNTOSY;
5535:     RSY[31] := PACKEDSY; RSY[32] := PROGRAMSY; RSY[33] := FUNCTSY;
5536:     RSY[34] := PROCSY;
5537:     SSY[#+#] := ADDOP; SSY[#-#] := ADDOP; SSY[#*#] := MULOP;
5538:     SSY[#/#] := MULOP; SSY[#(#] := LPARENT; SSY[#)#] := RPARENT;
5539:     SSY[#$#] := OTHERSY; SSY[#=#] := RELOP; SSY[# #] := OTHERSY;
5540:     SSY[#,#] := COMMA; SSY[#.#] := PERIOD; SSY[####] := OTHERSY;
5541:     SSY[#[#] := LBRACK; SSY[#]#] := RBRACK; SSY[#:#] := COLON;
5542:     SSY[#'#] := ARROW; SSY[#;#] := SEMICOLON;
5543:     SSY[#<#] := RELOP; SSY[#>#] := RELOP;
5544:    END (*SYMBOLS*) ;
```

```
5545:
5546:    PROCEDURE OPRTORS;
5547:     VAR I: INTEGER; CH: CHAR;
5548:    BEGIN
5549:     FOR I := 1 TO RESWORDS DO ROP[I] := NOOP;
5550:     ROP[5] := INOP; ROP[10] := IDIV; ROP[11] := IMOD;
5551:     ROP[6] := OROP; ROP[13] := ANDOP;
5552:     SOP[#(#] := NOOP; SOP[#)#] := NOOP; SOP[#$#] := NOOP;
5553:     SOP[# #] := NOOP; SOP[#,#] := NOOP; SOP[#.#] := NOOP;
5554:     SOP[#[#] := NOOP; SOP[#]#] := NOOP; SOP[#:#] := NOOP;
5555:     SOP[#'#] := NOOP; SOP[#;#] := NOOP; SOP[####] := NOOP;
5556:     SOP[#+#] := PLUS;SOP[#-#] := MINUS;SOP[#*#] := MUL;SOP[#/#] := RDIV;
5557:     SOP[#=#] := EQOP;SOP[#<#] := LTOP;SOP[#>#] := GTOP;
5558:    END (*OPRTORS*) ;
5559:
5560:   BEGIN (*INITTABLES*)
5561:    RESERVEDWORDS; SYMBOLS; OPRTORS;
5562:   END (*INITTABLES*) ;
5563:
5564:  BEGIN
5565:
5566:   (*INITIALIZE:*)
5567:   (*************)
5568:   INITSCALARS; INITSETS; INITTABLES;
5569:
5570:   (*ENTER STANDARD NAMES AND STANDARD TYPES:*)
5571:   (*****************************************)
5572:
5573:   LEVEL := 0; TOP := 0;
5574:   WITH DISPLAY[0] DO
5575:    BEGIN FNAME := NIL; OCCUR := BLCK; PFNAME := NIL END;
5576:   STDTYPENTRIES; STDNAMES; STDNAMENTRIES; ENTERUNDECL;
5577:   TOP := 1; LEVEL := 1;
5578:   WITH DISPLAY[1] DO
5579:    BEGIN FNAME := NIL; OCCUR := BLCK; PFNAME := NIL END;
5580:
5581:   (*COMPILE:*)
5582:   (**********)
5583:
5584:   INSYMBOL;
5585:   PROGRAMME(BLOCKBEGSYS+STATBEGSYS-[CASESY]);
5586:
5587:   TERMINATED := TRUE;
5588:   ENDOFLINE;
5589:   WRITELN(OUTPUT); WRITELN(OUTPUT);
5590:   WHILE NOT EOF(INPUT) DO READ(INPUT,CH)
5591:  END .

THAT'S ALL FOLKS!      LINES: 5591   CHARACTERS: 185464
```