```
0001:  (*$L+*)
0002:  (*$I XCOMP:A.TEXT *)
0003:
0004:  (*$U-*)
0005:  PROGRAM PASCALSYSTEM;
0006:
0007:  (*********************************************)
0008:  (*                                           *)
0009:  (*        UCSD  PASCAL  COMPILER             *)
0010:  (*                                           *)
0011:  (*    BASED ON ZURICH P2 PORTABLE            *)
0012:  (*    COMPILER, EXTENSIVLY                   *)
0013:  (*    MODIFIED BY ROGER T. SUMNER            *)
0014:  (*    1976..1977                             *)
0015:  (*                                           *)
0016:  (*    INSTITUTE FOR INFORMATION SYSTEMS      *)
0017:  (*    UC SAN DIEGO, LA JOLLA, CA             *)
0018:  (*                                           *)
0019:  (*    KENNETH L. BOWLES, DIRECTOR            *)
0020:  (*                                           *)
0021:  (*    THIS SOFTWARE IS THE PROPERTY OF THE   *)
0022:  (*  REGENTS OF THE UNIVERSITY OF CALIFORNIA.  *)
0023:  (*                                           *)
0024:  (*********************************************)
0025:
0026:  TYPE PHYLE = FILE;
0027:       INFOREC = RECORD
0028:                   WORKSYM,WORKCODE: ^PHYLE;
0029:                   ERRSYM,ERRBLK,ERRNUM: INTEGER;
0030:                   STUPID: BOOLEAN
0031:                 END;
0032:
0033:  PROGRAM PROCEDURE USERPROGRAM;
0034:  BEGIN END (*USERPROGRAM*) ;
0035:
0036:  PROGRAM PROCEDURE COMPILER(VAR USERINFO: INFOREC);
0037:
0038:  CONST DISPLIMIT = 12; MAXLEVEL = 8; MAXADDR = 28000;
0039:        INTSIZE = 1; REALSIZE = 2; BITSPERWD = 16;
0040:        CHARSIZE = 1; BOOLSIZE = 1; PTRSIZE = 1;
0041:        FILESIZE = 300; NILFILESIZE = 34; BITSPERCHR = 8; CHRSPERWD = 2;
0042:        STRINGSIZE = 0; STRGLGTH = 255; MAXINT = 32767;
0043:        DEFSTRGLGTH = 80; LCAFTERMARKSTACK = 1;
0044:        EOL = 13; MAXCURSOR = 1023; MAXCODE = 1299;
0045:        MAXJTAB = 24; MAXSEG = 15; MAXPROCNUM = 149;
0046:
0047:  TYPE
0048:                                              (*BASIC SYMBOLS*)
0049:
0050:       SYMBOL =  (IDENT,COMMA,COLON,SEMICOLON,LPARENT,RPARENT,DOSY,TOSY,
0051:                  DOWNTOSY,ENDSY,UNTILSY,OFSY,THENSY,ELSESY,BECOMES,LBRACK,
0052:                  RBRACK,ARROW,PERIOD,BEGINSY,IFSY,CASESY,REPEATSY,WHILESY,
0053:                  FORSY,WITHSY,GOTOSY,LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,
0054:                  FUNCSY,PROGSY,FORWARDSY,INTCONST,REALCONST,STRINGCONST,
0055:                  NOTSY,MULOP,ADDOP,RELOP,SETSY,PACKEDSY,ARRAYSY,RECORDSY,
0056:                  FILESY,OTHERSY);
0057:
0058:
0059:       OPERATOR = (MUL,RDIV,ANDOP,IDIV,IMOD,PLUS,MINUS,OROP,LTOP,LEOP,
0060:                   GEOP,GTOP,NEOP,EQOP,INOP,NOOP);
0061:
0062:       SETOFSYS = SET OF SYMBOL;
0063:
0064:                                              (*CONSTANTS*)
0065:       CSTCLASS = (REEL,PSET,STRG,TRIX);
0066:       CSP = ^ CONSTREC;
```

```
0067:       CONSTREC = RECORD CASE CCLASS: CSTCLASS OF
0068:                           TRIX: (CSTVAL: ARRAY [1..8] OF INTEGER);
0069:                           REEL: (RVAL: REAL);
0070:                           PSET: (PVAL: SET OF 0..127);
0071:                           STRG: (SLGTH: 0..STRGLGTH;
0072:                                  SVAL: PACKED ARRAY [1..STRGLGTH] OF CHAR)
0073:                         END;
0074:
0075:       VALU = RECORD CASE BOOLEAN OF
0076:                       TRUE:  (IVAL: INTEGER);
0077:                       FALSE: (VALP: CSP)
0078:                     END;
0079:
0080:                                                    (*DATA STRUCTURES*)
0081:       BITRANGE = 0..BITSPERWD; OPRANGE = 0..127;
0082:       CURSRANGE = 0..MAXCURSOR; PROCRANGE = 0..MAXPROCNUM;
0083:       LEVRANGE = 0..MAXLEVEL; ADDRRANGE = 0..MAXADDR;
0084:       JTABRANGE = 0..MAXJTAB; SEGRANGE = 0..MAXSEG;
0085:       DISPRANGE = 0..DISPLIMIT;
0086:
0087:       STRUCTFORM = (SCALAR,SUBRANGE,POINTER,POWER,ARRAYS,
0088:                     RECORDS,FILES,TAGFLD,VARIANT);
0089:
0090:       DECLKIND = (STANDARD,DECLARED,SPECIAL);
0091:
0092:       STP = ^ STRUCTURE; CTP = ^ IDENTIFIER;
0093:
0094:       STRUCTURE = RECORD
0095:                     SIZE: ADDRRANGE;
0096:                     CASE FORM: STRUCTFORM OF
0097:                       SCALAR:   (CASE SCALKIND: DECLKIND OF
0098:                                    DECLARED: (FCONST: CTP));
0099:                       SUBRANGE: (RANGETYPE: STP; MIN,MAX: VALU);
0100:                       POINTER:  (ELTYPE: STP);
0101:                       POWER:    (ELSET: STP);
0102:                       ARRAYS:   (AELTYPE,INXTYPE: STP;
0103:                                  CASE AISPACKD:BOOLEAN OF
0104:                                    TRUE: (ELSPERWD,ELWIDTH: BITRANGE;
0105:                                           CASE AISSTRNG: BOOLEAN OF
0106:                                            TRUE:(MAXLENG: 1..STRGLGTH)));
0107:                       RECORDS:  (FSTFLD: CTP; RECVAR: STP);
0108:                       FILES:    (FILTYPE: STP);
0109:                       TAGFLD:   (TAGFIELDP: CTP; FSTVAR: STP);
0110:                       VARIANT:  (NXTVAR,SUBVAR: STP; VARVAL: VALU)
0111:                     END;
0112:
0113:                                                          (*NAMES*)
0114:       IDCLASS = (TYPES,KONST,VARS,FIELD,PROC,FUNC);
0115:       SETOFIDS = SET OF IDCLASS;
0116:       IDKIND = (ACTUAL,FORMAL);
0117:       ALPHA = PACKED ARRAY [1..8] OF CHAR;
0118:
0119:       IDENTIFIER = RECORD
0120:                      NAME: ALPHA; LLINK, RLINK: CTP;
0121:                      IDTYPE: STP; NEXT: CTP;
0122:                      CASE KLASS: IDCLASS OF
0123:                        KONST: (VALUES: VALU);
0124:                        VARS:  (VKIND: IDKIND; VLEV: LEVRANGE;
0125:                                VADDR: ADDRRANGE);
0126:                        FIELD: (FLDADDR: ADDRRANGE;
0127:                                CASE FISPACKD: BOOLEAN OF
0128:                                  TRUE: (FLDRBIT,FLDWIDTH: BITRANGE));
0129:                        PROC,
0130:                        FUNC:  (CASE PFDECKIND: DECLKIND OF
0131:                                  SPECIAL:  (KEY: 1..23);
0132:                                  STANDARD: (CSPNUM: 1..40);
```

```
0133:                                DECLARED: (PFLEV: LEVRANGE;
0134:                                           PFNAME: PROCRANGE;
0135:                                           PFSEG: SEGRANGE;
0136:                                           CASE PFKIND: IDKIND OF
0137:                                             ACTUAL: (LOCALLC: ADDRRANGE;
0138:                                                      FORWDECL,
0139:                                                      INSCOPE: BOOLEAN)))
0140:                      END;
0141:
0142:
0143:       WHERE = (BLCK,CREC,VREC,REC);
0144:
0145:                                                 (*EXPRESSIONS*)
0146:       ATTRKIND = (CST,VARBL,EXPR);
0147:       VACCESS = (DRCT,INDRCT,PACKD,MULTI,BYTE);
0148:
0149:       ATTR = RECORD TYPTR: STP;
0150:                CASE KIND: ATTRKIND OF
0151:                  CST:   (CVAL: VALU);
0152:                  VARBL: (CASE ACCESS: VACCESS OF
0153:                            DRCT:   (VLEVEL: LEVRANGE; DPLMT: ADDRRANGE);
0154:                            INDRCT: (IDPLMT: ADDRRANGE))
0155:              END;
0156:
0157:       TESTP = ^ TESTPOINTER;
0158:       TESTPOINTER = RECORD
0159:                       ELT1,ELT2 : STP;
0160:                       LASTTESTP : TESTP
0161:                     END;
0162:
0163:                                                    (*LABELS*)
0164:       LBP = ^ CODELABEL;
0165:       CODELABEL = RECORD
0166:                     CASE DEFINED: BOOLEAN OF
0167:                       FALSE: (REFLIST: ADDRRANGE);
0168:                       TRUE:  (OCCURIC: ADDRRANGE; JTABINX: JTABRANGE)
0169:                   END;
0170:
0171:       LABELP = ^ USERLABEL;
0172:       USERLABEL = RECORD
0173:                     LABVAL: INTEGER;
0174:                     NEXTLAB: LABELP;
0175:                     CODELBP: LBP
0176:                   END;
0177:
0178:       CODEARRAY = PACKED ARRAY [0..MAXCODE] OF CHAR;
0179:       SYMBUFARRAY = PACKED ARRAY [CURSRANGE] OF CHAR;
0180:
0181:  (*-------------------------------------------------------------------*)
0182:
0183:  VAR
0184:
0185:      CODEP: ^ CODEARRAY;             (*CODE BUFFER UNTIL WRITEOUT*)
0186:      SYMBUFP: ^ SYMBUFARRAY;         (*SYMBOLIC BUFFER...ASCII OR CODED*)
0187:
0188:      GATTR: ATTR;                    (*DESCRIBES CURRENT EXPRESSION*)
0189:      VAL: VALU;                      (*VALUE OF LAST CONSTANT*)
0190:
0191:      DISX,                           (*LEVEL OF LAST ID SEARCHED*)
0192:      TOP: DISPRANGE;                 (*TOP OF DISPLAY*)
0193:                                      (*SCANNER GLOBALS...NEXT FOUR VARS*)
0194:                                      (*MUST BE IN THIS ORDER FOR IDSEARCH*)
0195:      SYMCURSOR: CURSRANGE;           (*CURRENT SCANNING INDEX IN SYMBUFP^*)
0196:      SY: SYMBOL;                     (*SYMBOL FOUND BY INSYMBOL*)
0197:      OP: OPERATOR;                   (*CLASSIFICATION OF LAST SYMBOL*)
0198:      ID: ALPHA;                      (*LAST IDENTIFIER FOUND*)
```

```
0199:
0200:      LGTH: INTEGER;                  (*LENGTH OF LAST STRING CONSTANT*)
0201:
0202:      LCMAX,LC,IC: ADDRRANGE;         (*LOCATION AND INSTRUCT COUNTERS*)
0203:
0204:                                      (*SWITCHES:*)
0205:
0206:      PRTERR,GOTOOK,RANGECHECK,CODEINSEG,IOCHECK,
0207:      LIST,TEST,SYSCOMP,DP,INCLUDING: BOOLEAN;
0208:
0209:                                      (*POINTERS:*)
0210:      INTPTR,REALPTR,CHARPTR,BOOLPTR,
0211:      TEXTPTR,NILPTR,STRGPTR: STP;    (*POINTERS TO STANDARD IDS*)
0212:
0213:      UTYPPTR,UCSTPTR,UVARPTR,
0214:      UFLDPTR,UPRCPTR,UFCTPTR,        (*POINTERS TO UNDECLARED IDS*)
0215:      INPUTPTR,OUTPUTPTR,
0216:      OUTERBLOCK,FWPTR: CTP;
0217:
0218:      GLOBTESTP: TESTP;                (*LAST TESTPOINTER*)
0219:
0220:      LEVEL: LEVRANGE;                (*CURRENT STATIC LEVEL*)
0221:
0222:      SEG,NEXTSEG: SEGRANGE;          (*CURRENT SEGMENT #*)
0223:      SEGINX: INTEGER;                (*CURRENT INDEX IN SEGMENT*)
0224:      SCONST: CSP;                    (*INSYMBOL STRING RESULTS*)
0225:
0226:      LOWTIME,LINEINFO,SCREENDOTS,STARTDOTS,SYMBLK: INTEGER;
0227:      LINESTART: CURSRANGE;
0228:
0229:      CURPROC,NEXTPROC: PROCRANGE;     (*PROCEDURE NUMBER ASSIGNMENT*)
0230:
0231:      CONSTBEGSYS,SIMPTYPEBEGSYS,TYPEBEGSYS,BLOCKBEGSYS,
0232:      SELECTSYS,FACBEGSYS,STATBEGSYS,TYPEDELS: SETOFSYS;
0233:
0234:      DISPLAY: ARRAY [DISPRANGE] OF
0235:                  RECORD
0236:                    FNAME: CTP;
0237:                    CASE OCCUR: WHERE OF
0238:                      BLCK: (FFILE: CTP; FLABEL: LABELP);
0239:                      CREC: (CLEV: LEVRANGE; CDSPL: ADDRRANGE);
0240:                      VREC: (VDSPL: ADDRRANGE)
0241:                    END;
0242:
0243:      PROCTABLE: ARRAY [PROCRANGE] OF INTEGER;
0244:
0245:      SEGTABLE: ARRAY [SEGRANGE] OF
0246:                  RECORD
0247:                    DISKADDR,CODELENG: INTEGER;
0248:                    SEGNAME: ALPHA
0249:                  END (*SEGTABLE*) ;
0250:
0251:      NEXTJTAB: JTABRANGE;
0252:      JTAB: ARRAY [JTABRANGE] OF INTEGER;
0253:
0254:      OLDSYMBLK: INTEGER;
0255:      OLDSYMCURSOR: CURSRANGE;
0256:      INCLFILE: FILE;
0257:
0258:      CURBYTE, CURBLK: INTEGER;
0259:      DISKBUF: PACKED ARRAY [0..511] OF CHAR;
0260:
0261:  (*-------------------------------------------------------------------*)
0262:
0263:  PROCEDURE INSYMBOL;
0264:    FORWARD;
```

```
0265:
0266:  PROCEDURE ERROR(ERRORNUM: INTEGER);
0267:    FORWARD;
0268:
0269:  PROCEDURE ENTERID(FCP: CTP);
0270:    FORWARD;
0271:
0272:  PROCEDURE GETNEXTPAGE;
0273:    FORWARD;
0274:
0275:  PROGRAM PROCEDURE COMPINIT;
0276:
0277:    PROCEDURE ENTSTDTYPES;
0278:      VAR SP: STP;
0279:    BEGIN
0280:      NEW(INTPTR,SCALAR,STANDARD);
0281:      WITH INTPTR^ DO
0282:        BEGIN SIZE := INTSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
0283:      NEW(REALPTR,SCALAR,STANDARD);
0284:      WITH REALPTR^ DO
0285:        BEGIN SIZE := REALSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
0286:      NEW(CHARPTR,SCALAR,STANDARD);
0287:      WITH CHARPTR^ DO
0288:        BEGIN SIZE := CHARSIZE; FORM := SCALAR; SCALKIND := STANDARD END;
0289:      NEW(BOOLPTR,SCALAR,DECLARED);
0290:      WITH BOOLPTR^ DO
0291:        BEGIN SIZE := BOOLSIZE; FORM := SCALAR; SCALKIND := DECLARED END;
0292:      NEW(NILPTR,POINTER);
0293:      WITH NILPTR^ DO
0294:        BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
0295:      NEW(TEXTPTR,FILES);
0296:      WITH TEXTPTR^ DO
0297:        BEGIN SIZE := FILESIZE+CHARSIZE; FORM := FILES; FILTYPE := CHARPTR END;
0298:      NEW(STRGPTR,ARRAYS,TRUE,TRUE);
0299:      WITH STRGPTR^ DO
0300:        BEGIN FORM := ARRAYS; SIZE := (DEFSTRGLGTH + CHRSPERWD) DIV CHRSPERWD;
0301:          AISPACKD := TRUE; AISSTRNG := TRUE; INXTYPE := INTPTR;
0302:          ELWIDTH := BITSPERCHR; ELSPERWD := CHRSPERWD;
0303:          AELTYPE := CHARPTR; MAXLENG := DEFSTRGLGTH;
0304:        END
0305:    END (*ENTSTDTYPES*) ;
0306:
0307:    PROCEDURE ENTSTDNAMES;
0308:      VAR CP,CP1: CTP; I: INTEGER;
0309:    BEGIN
0310:      NEW(CP,TYPES);
0311:      WITH CP^ DO
0312:        BEGIN NAME := 'INTEGER '; IDTYPE := INTPTR; KLASS := TYPES END;
0313:      ENTERID(CP);
0314:      NEW(CP,TYPES);
0315:      WITH CP^ DO
0316:        BEGIN NAME := 'REAL    '; IDTYPE := REALPTR; KLASS := TYPES END;
0317:      ENTERID(CP);
0318:      NEW(CP,TYPES);
0319:      WITH CP^ DO
0320:        BEGIN NAME := 'CHAR    '; IDTYPE := CHARPTR; KLASS := TYPES END;
0321:      ENTERID(CP);
0322:      NEW(CP,TYPES);
0323:      WITH CP^ DO
0324:        BEGIN NAME := 'BOOLEAN '; IDTYPE := BOOLPTR; KLASS := TYPES END;
0325:      ENTERID(CP);
0326:      NEW(CP,TYPES);
0327:      WITH CP^ DO
0328:        BEGIN NAME := 'STRING  '; IDTYPE := STRGPTR; KLASS := TYPES END;
0329:      ENTERID(CP);
0330:      NEW(CP,TYPES);
```

```
0331:      WITH CP^ DO
0332:        BEGIN NAME := 'TEXT    '; IDTYPE := TEXTPTR; KLASS := TYPES END;
0333:      ENTERID(CP);
0334:      NEW(INPUTPTR,VARS);
0335:      WITH INPUTPTR^ DO
0336:        BEGIN NAME := 'INPUT   '; IDTYPE := TEXTPTR; KLASS := VARS;
0337:          VKIND := FORMAL; VLEV := 0; VADDR := 2
0338:        END;
0339:      ENTERID(INPUTPTR);
0340:      NEW(OUTPUTPTR,VARS);
0341:      WITH OUTPUTPTR^ DO
0342:        BEGIN NAME := 'OUTPUT  '; IDTYPE := TEXTPTR; KLASS := VARS;
0343:          VKIND := FORMAL; VLEV := 0; VADDR := 3
0344:        END;
0345:      ENTERID(OUTPUTPTR);
0346:      NEW(CP,VARS);
0347:      WITH CP^ DO
0348:        BEGIN NAME := 'KEYBOARD'; IDTYPE := TEXTPTR; KLASS := VARS;
0349:          VKIND := FORMAL; VLEV := 0; VADDR := 4
0350:        END;
0351:      ENTERID(CP);
0352:      CP1 := NIL;
0353:      FOR I := 0 TO 1 DO
0354:        BEGIN NEW(CP,KONST);
0355:          WITH CP^ DO
0356:            BEGIN IDTYPE := BOOLPTR;
0357:              IF I = 0 THEN NAME := 'FALSE   '
0358:              ELSE NAME := 'TRUE    ';
0359:              NEXT := CP1; VALUES.IVAL := I; KLASS := KONST
0360:            END;
0361:          ENTERID(CP); CP1 := CP
0362:        END;
0363:      BOOLPTR^.FCONST := CP;
0364:      NEW(CP,KONST);
0365:      WITH CP^ DO
0366:        BEGIN NAME := 'NIL     '; IDTYPE := NILPTR;
0367:          NEXT := NIL; VALUES.IVAL := 0; KLASS := KONST
0368:        END;
0369:      ENTERID(CP);
0370:    END (*ENTSTDNAMES*) ;
0371:
0372:    PROCEDURE ENTUNDECL;
0373:    BEGIN
0374:      NEW(UTYPPTR,TYPES);
0375:      WITH UTYPPTR^ DO
0376:        BEGIN NAME := '        '; IDTYPE := NIL; KLASS := TYPES END;
0377:      NEW(UCSTPTR,KONST);
0378:      WITH UCSTPTR^ DO
0379:        BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
0380:          VALUES.IVAL := 0; KLASS := KONST
0381:        END;
0382:      NEW(UVARPTR,VARS);
0383:      WITH UVARPTR^ DO
0384:        BEGIN NAME := '        '; IDTYPE := NIL; VKIND := ACTUAL;
0385:          NEXT := NIL; VLEV := 0; VADDR := 0; KLASS := VARS
0386:        END;
0387:      NEW(UFLDPTR,FIELD);
0388:      WITH UFLDPTR^ DO
0389:        BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
0390:          FLDADDR := 0; KLASS := FIELD
0391:        END;
0392:      NEW(UPRCPTR,PROC,DECLARED,ACTUAL);
0393:      WITH UPRCPTR^ DO
0394:        BEGIN NAME := '        '; IDTYPE := NIL; FORWDECL := FALSE;
0395:          NEXT := NIL; INSCOPE := FALSE; LOCALLC := 0;
0396:          PFLEV := 0; PFNAME := 0; PFSEG := 0;
```

```
0397:          KLASS := PROC; PFDECKIND := DECLARED; PFKIND := ACTUAL
0398:        END;
0399:      NEW(UFCTPTR,FUNC,DECLARED,ACTUAL);
0400:      WITH UFCTPTR^ DO
0401:        BEGIN NAME := '        '; IDTYPE := NIL; NEXT := NIL;
0402:          FORWDECL := FALSE; INSCOPE := FALSE; LOCALLC := 0;
0403:          PFLEV := 0; PFNAME := 0; PFSEG := 0;
0404:          KLASS := FUNC; PFDECKIND := DECLARED; PFKIND := ACTUAL
0405:        END
0406:    END (*ENTUNDECL*) ;
0407:
0408:    PROCEDURE ENTSPCPROCS;
0409:      VAR LCP: CTP; I: INTEGER; ISFUNC: BOOLEAN;
0410:          NA: ARRAY [1..42] OF ALPHA;
0411:    BEGIN
0412:      NA[ 1] := 'READ    '; NA[ 2] := 'READLN  '; NA[ 3] := 'WRITE   ';
0413:      NA[ 4] := 'WRITELN '; NA[ 5] := 'EOF     '; NA[ 6] := 'EOLN    ';
0414:      NA[ 7] := 'PRED    '; NA[ 8] := 'SUCC    '; NA[ 9] := 'ORD     ';
0415:      NA[10] := 'SQR     '; NA[11] := 'ABS     '; NA[12] := 'NEW     ';
0416:      NA[13] := 'UNITREAD'; NA[14] := 'UNITWRIT'; NA[15] := 'CONCAT  ';
0417:      NA[16] := 'LENGTH  '; NA[17] := 'INSERT  '; NA[18] := 'DELETE  ';
0418:      NA[19] := 'COPY    '; NA[20] := 'POS     '; NA[21] := 'MOVELEFT';
0419:      NA[22] := 'MOVERIGH'; NA[23] := 'EXIT    '; NA[24] := 'IDSEARCH';
0420:      NA[25] := 'TREESEAR'; NA[26] := 'TIME    '; NA[27] := 'FILLCHAR';
0421:      NA[28] := 'OPENNEW '; NA[29] := 'OPENOLD '; NA[30] := 'REWRITE ';
0422:      NA[31] := 'CLOSE   '; NA[32] := 'SEEK    '; NA[33] := 'RESET   ';
0423:      NA[34] := 'GET     '; NA[35] := 'PUT     '; NA[36] := 'SCAN    ';
0424:      NA[37] := 'BLOCKREA'; NA[38] := 'BLOCKWRI'; NA[39] := 'DRAWLINE';
0425:      NA[40] := 'PAGE    '; NA[41] := 'SIZEOF  '; NA[42] := 'DRAWBLOC';
0426:      FOR I := 1 TO 42 DO
0427:        BEGIN ISFUNC := I IN [5,6,7,8,9,10,11,15,16,19,20,25,36,37,38,41];
0428:          IF ISFUNC THEN NEW(LCP,FUNC,SPECIAL)
0429:          ELSE NEW(LCP,PROC,SPECIAL);
0430:          WITH LCP^ DO
0431:            BEGIN NAME := NA[I]; NEXT := NIL; IDTYPE := NIL;
0432:              IF ISFUNC THEN KLASS := FUNC ELSE KLASS := PROC;
0433:              PFDECKIND := SPECIAL; KEY := I
0434:            END;
0435:          ENTERID(LCP)
0436:        END
0437:      END (*ENTSPCPROCS*) ;
0438:
0439:    PROCEDURE ENTSTDPROCS;
0440:      VAR LCP,PARAM: CTP; LSP,FTYPE: STP; I: INTEGER; ISPROC: BOOLEAN;
0441:          NA: ARRAY [1..19] OF ALPHA;
0442:    BEGIN
0443:      NA[ 1] := 'ODD     '; NA[ 2] := 'CHR     '; NA[ 3] := 'TRUNC   ';
0444:      NA[ 4] := 'ROUND   '; NA[ 5] := 'SIN     '; NA[ 6] := 'COS     ';
0445:      NA[ 7] := 'LOG     '; NA[ 8] := 'ATAN    '; NA[ 9] := 'LN      ';
0446:      NA[10] := 'EXP     '; NA[11] := 'SQRT    '; NA[12] := 'MARK    ';
0447:      NA[13] := 'RELEASE '; NA[14] := 'IORESULT'; NA[15] := 'UNITBUSY';
0448:      NA[16] := 'PWROFTEN'; NA[17] := 'UNITWAIT'; NA[18] := 'UNITCLEA';
0449:      NA[19] := 'HALT    ';
0450:      FOR I := 1 TO 19 DO
0451:        BEGIN ISPROC := I IN [12,13,17,18,19];
0452:          CASE I OF
0453:            1:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,VARS);
0454:                WITH PARAM^ DO
0455:                  BEGIN IDTYPE := INTPTR; VKIND := ACTUAL END;
0456:                END;
0457:            2:  FTYPE := CHARPTR;
0458:            3:  BEGIN FTYPE := INTPTR; NEW(PARAM,VARS);
0459:                WITH PARAM^ DO
0460:                  BEGIN IDTYPE := REALPTR; VKIND := ACTUAL END;
0461:                END;
0462:            5:  FTYPE := REALPTR;
```

```
0463:           12:  BEGIN FTYPE := NIL; NEW(PARAM,VARS); NEW(LSP,POINTER);
0464:                WITH LSP^ DO
0465:                  BEGIN SIZE := PTRSIZE; FORM := POINTER; ELTYPE := NIL END;
0466:                WITH PARAM^ DO
0467:                  BEGIN IDTYPE := LSP; VKIND := FORMAL END;
0468:                END;
0469:           14:  BEGIN FTYPE := INTPTR; PARAM := NIL END;
0470:           15:  BEGIN FTYPE := BOOLPTR; NEW(PARAM,VARS);
0471:                WITH PARAM^ DO
0472:                  BEGIN IDTYPE := INTPTR; VKIND := ACTUAL END;
0473:                END;
0474:           16:  FTYPE := REALPTR;
0475:           17:  FTYPE := NIL;
0476:           19:  BEGIN FTYPE := NIL; PARAM := NIL END;
0477:         END (*PARAM AND TYPE CASES*) ;
0478:         IF ISPROC THEN NEW(LCP,PROC,STANDARD)
0479:         ELSE NEW(LCP,FUNC,STANDARD);
0480:         WITH LCP^ DO
0481:           BEGIN NAME := NA[I]; PFDECKIND := STANDARD; CSPNUM := I + 20;
0482:             IF ISPROC THEN KLASS := PROC ELSE KLASS := FUNC;
0483:             IF PARAM <> NIL THEN
0484:               WITH PARAM^ DO
0485:                 BEGIN KLASS := VARS; NEXT := NIL END;
0486:             IDTYPE := FTYPE; NEXT := PARAM
0487:           END;
0488:         ENTERID(LCP)
0489:       END
0490:     END (*ENTSTDPROCS*) ;
0491:
0492:   PROCEDURE INITSCALARS;
0493:   BEGIN FWPTR := NIL; GLOBTESTP := NIL;
0494:     LINESTART := 0; LINEINFO := LCAFTERMARKSTACK; LIST := FALSE;
0495:     SYMBLK := 2; SCREENDOTS := 0; STARTDOTS := 0;
0496:     FOR SEG := 0 TO MAXSEG DO
0497:       WITH SEGTABLE[SEG] DO
0498:         BEGIN DISKADDR := 0; CODELENG := 0; SEGNAME := '        ' END;
0499:     LC := LCAFTERMARKSTACK; IOCHECK := TRUE; DP := TRUE;
0500:     SEGINX := 0; NEXTJTAB := 1; NEXTPROC := 2; CURPROC := 1;
0501:     NEW(SCONST); NEW(SYMBUFP); NEW(CODEP);
0502:     SEG := 1; NEXTSEG := 10; CURBLK := 1; CURBYTE := 0;
0503:     GOTOOK := FALSE; RANGECHECK := TRUE; SYSCOMP := FALSE;
0504:     CODEINSEG := FALSE; PRTERR := TRUE; INCLUDING := FALSE
0505:   END (*INITSCALARS*) ;
0506:
0507:   PROCEDURE INITSETS;
0508:   BEGIN
0509:     CONSTBEGSYS := [ADDOP,INTCONST,REALCONST,STRINGCONST,IDENT];
0510:     SIMPTYPEBEGSYS := [LPARENT] + CONSTBEGSYS;
0511:     TYPEBEGSYS := [ARROW,PACKEDSY,ARRAYSY,RECORDSY,SETSY,FILESY]
0512:                   + SIMPTYPEBEGSYS;
0513:     TYPEDELS := [ARRAYSY,RECORDSY,SETSY,FILESY];
0514:     BLOCKBEGSYS := [LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,FUNCSY,PROGSY,BEGINSY];
0515:     SELECTSYS := [ARROW,PERIOD,LBRACK];
0516:     FACBEGSYS := [INTCONST,REALCONST,STRINGCONST,IDENT,LPARENT,LBRACK,NOTSY];
0517:     STATBEGSYS := [BEGINSY,GOTOSY,IFSY,WHILESY,REPEATSY,FORSY,WITHSY,CASESY]
0518:   END (*INITSETS*) ;
0519:
0520: BEGIN (*COMPINIT*)
0521:   INITSCALARS; INITSETS;
0522:   LEVEL := 0; TOP := 0;
0523:   WITH DISPLAY[0] DO
0524:     BEGIN FNAME := NIL; FFILE := NIL; FLABEL := NIL; OCCUR := BLCK END;
0525:   ENTSTDTYPES;   ENTSTDNAMES;   ENTUNDECL;
0526:   ENTSPCPROCS;   ENTSTDPROCS;
0527:   GETNEXTPAGE;
0528:   UNITWRITE(3,PROCTABLE[-1200],35);
```

```
0529:    FOR IC := 1 TO 7 DO WRITELN(OUTPUT);
0530:    WRITELN(OUTPUT,'PASCAL compilation');
0531:    WRITE(OUTPUT,'<   0>');
0532:    INSYMBOL;
0533:    IF SYSCOMP THEN
0534:      BEGIN OUTERBLOCK := NIL; SEG := 0; NEXTSEG := 1 END
0535:    ELSE
0536:      BEGIN TOP := 1; LEVEL := 1;
0537:        WITH DISPLAY[1] DO
0538:          BEGIN FNAME := NIL; FFILE := NIL;
0539:            FLABEL := NIL; OCCUR := BLCK
0540:          END;
0541:        LC := LC+2; (*KEEP STACK STRAIGHT FOR NOW*)
0542:        NEW(OUTERBLOCK,PROC,DECLARED,ACTUAL);
0543:        WITH OUTERBLOCK^ DO
0544:          BEGIN NEXT := NIL; LOCALLC := LC;
0545:            NAME := 'PROGRAM '; IDTYPE := NIL; KLASS := PROC;
0546:            PFDECKIND := DECLARED; PFLEV := 0; PFNAME := 1; PFSEG := SEG;
0547:            PFKIND := ACTUAL; FORWDECL := FALSE; INSCOPE := TRUE
0548:          END
0549:      END;
0550:    IF SY = PROGSY THEN
0551:      BEGIN INSYMBOL;
0552:        IF SY = IDENT THEN
0553:          BEGIN SEGTABLE[SEG].SEGNAME := ID;
0554:            IF OUTERBLOCK <> NIL THEN OUTERBLOCK^.NAME := ID;
0555:          END
0556:        ELSE ERROR(2); INSYMBOL;
0557:        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
0558:      END
0559:  END (*COMPINIT*) ;
0560:
0561:  (*$I XCOMP:B.TEXT *)
0562:
0563:  PROCEDURE ERROR(*ERRORNUM: INTEGER*);
0564:    VAR CH: CHAR;
0565:  BEGIN
0566:    WITH USERINFO DO
0567:      IF (ERRSYM <> SYMCURSOR) OR (ERRBLK <> SYMBLK) THEN
0568:        BEGIN
0569:          ERRSYM := SYMCURSOR; ERRBLK := SYMBLK;
0570:          ERRNUM := ERRORNUM;
0571:          IF STUPID THEN EXIT(COMPILER);
0572:          WRITELN(OUTPUT); CH := ' ';
0573:          WRITE(OUTPUT,SYMBUFP^:SYMCURSOR)
0574:          WRITELN(OUTPUT,' <<<<' Error # ',ERRORNUM:0);
0575:          WRITE(OUTPUT,'Hit <SPACE> to continue');
0576:          REPEAT UNITREAD(2,CH,1);
0577:          UNTIL (CH = ' ') OR (CH = CHR(27));
0578:          IF (ERRORNUM > 400) OR (CH = CHR(27)) THEN EXIT(COMPILER);
0579:          WRITELN(OUTPUT);
0580:          WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
0581:        END
0582:  END (*ERROR*) ;
0583:
0584:  PROCEDURE GETNEXTPAGE;
0585:  BEGIN SYMCURSOR := 0;
0586:    IF INCLUDING THEN
0587:      IF BLOCKREAD(INCLFILE,SYMBUFP^,0,SYMBLK) = 0 THEN
0588:        BEGIN CLOSE(INCLFILE); INCLUDING := FALSE;
0589:          SYMBLK := OLDSYMBLK; SYMCURSOR := OLDSYMCURSOR;
0590:          LINESTART := SYMCURSOR   (*AT CR...WILL PRINT EXTRA LINE*)
0591:        END;
0592:    IF NOT INCLUDING THEN
0593:      IF BLOCKREAD(USERINFO.WORKSYM^,SYMBUFP^,2,SYMBLK) <> 2 THEN
0594:        ERROR(401);
```

```
0595:    SYMBLK := SYMBLK+2
0596:  END (*GETNEXTPAGE*) ;
0597:
0598:  PROCEDURE PRINTLINE;
0599:    VAR LPUNIT: INTEGER;
0600:        A: PACKED ARRAY [0..1] OF CHAR;
0601:
0602:    PROCEDURE WRITEINT(IVAL: INTEGER);
0603:      VAR I,IPOT: INTEGER; CH: CHAR; ZAP: BOOLEAN;
0604:          A: PACKED ARRAY [0..5] OF CHAR;
0605:    BEGIN ZAP := TRUE; IPOT := 10000; A[0] := ' ';
0606:      FOR I := 1 TO 5 DO
0607:        BEGIN
0608:          CH := CHR(IVAL DIV IPOT + ORD('0'));
0609:          IF I <> 5 THEN
0610:            IF ZAP THEN
0611:              IF CH = '0' THEN CH := ' '
0612:              ELSE ZAP := FALSE;
0613:          A[I] := CH;
0614:          IVAL := IVAL MOD IPOT;
0615:          IPOT := IPOT DIV 10
0616:        END;
0617:      UNITWRITE(LPUNIT,A,6)
0618:    END (*WRITEINT*) ;
0619:
0620:  BEGIN LPUNIT := 6; (*PRINTLINE*)
0621:    WRITEINT(SCREENDOTS); WRITEINT(CURPROC);
0622:    A[0] := ':';
0623:    IF DP THEN A[1] := 'D' ELSE A[1] := 'C';
0624:    UNITWRITE(LPUNIT,A,2); WRITEINT(LINEINFO);
0625:    A := '  '; UNITWRITE(LPUNIT,A,2); UNITWRITE(LPUNIT,A,2);
0626:    UNITWRITE(LPUNIT,SYMBUFP^[LINESTART],SYMCURSOR-LINESTART,,TRUE)
0627:  END (*PRINTLINE*) ;
0628:
0629:  PROCEDURE STARTINCL;
0630:  (*I APOLOGIZE FOR SUCH KLOODGE AS THIS BUT IT HAS TO
0631:    BE IN RIGHT NOW...*)
0632:  VAR TSTART,TLENG: INTEGER; TITLE: STRING[40];
0633:  BEGIN
0634:    TSTART := SYMCURSOR+2;
0635:    SYMCURSOR := SCAN(80,=CHR(EOL),SYMBUFP^[TSTART])+TSTART+1;
0636:    TLENG := SYMCURSOR-TSTART-3;
0637:    TITLE[0] := CHR(TLENG);
0638:    MOVELEFT(SYMBUFP^[TSTART],TITLE[1],TLENG);
0639:    OPENOLD(INCLFILE,TITLE);
0640:    IF IORESULT <> 0 THEN
0641:      BEGIN
0642:        OPENOLD(INCLFILE,CONCAT(TITLE,'.TEXT'));
0643:        IF IORESULT <> 0 THEN ERROR(403)
0644:      END;
0645:    SCREENDOTS := SCREENDOTS+1;
0646:    IF LIST THEN PRINTLINE;
0647:    INCLUDING := TRUE;
0648:    OLDSYMCURSOR := SYMCURSOR-1; (*POINT AT CR...PREVENT END PAGE BLOWUP*)
0649:    OLDSYMBLK := SYMBLK-2; (*SYMBLK IS NEXT TO READ...SAVE CUR PAGE#*)
0650:    SYMBLK := 2; GETNEXTPAGE; LINESTART := SYMCURSOR;
0651:    INSYMBOL; EXIT(INSYMBOL)  (*WEIRD, ISNT IT...*)
0652:  END (*STARTINCL*) ;
0653:
0654:  PROCEDURE INSYMBOL; (* COMPILER VERSION 3.4 06-NOV-76 *)
0655:    LABEL 1;
0656:    VAR LVP: CSP; X: INTEGER;
0657:
0658:  PROCEDURE CHECKEND;
0659:  BEGIN (* CHECKS FOR THE END OF THE PAGE *)
0660:    WRITE(OUTPUT,'.');
```

```
0661:    SCREENDOTS := SCREENDOTS+1;
0662:    SYMCURSOR := SYMCURSOR + 1;
0663:    IF (SCREENDOTS-STARTDOTS) MOD 50 = 0 THEN
0664:      BEGIN WRITELN(OUTPUT);
0665:        WRITE(OUTPUT,'<',SCREENDOTS:4,'>')
0666:      END
0667:    IF LIST THEN PRINTLINE;
0668:    IF SYMBUFP^[SYMCURSOR]=CHR(0) THEN GETNEXTPAGE
0669:    ELSE LINESTART := SYMCURSOR;
0670:    IF SYMBUFP^[SYMCURSOR] = CHR(16(*DLE*)) THEN
0671:      SYMCURSOR := SYMCURSOR+2
0672:    ELSE
0673:      BEGIN
0674:        SYMCURSOR := SYMCURSOR+SCAN(80,<>CHR(9),SYMBUFP^[SYMCURSOR]);
0675:        SYMCURSOR := SYMCURSOR+SCAN(80,<>' ',SYMBUFP^[SYMCURSOR])
0676:      END;
0677:    IF DP THEN LINEINFO := LC ELSE LINEINFO := IC
0678:  END;
0679:
0680:  PROCEDURE COMMENTER;
0681:    VAR CH,SW,DEL: CHAR;
0682:  BEGIN
0683:    SYMCURSOR := SYMCURSOR+2; (* POINT TO THE FIRST CH PAST "(*" *)
0684:    IF SYMBUFP^[SYMCURSOR]='$'
0685:      THEN
0686:        BEGIN
0687:          IF SYMBUFP^[SYMCURSOR+1] <> '*' THEN
0688:           REPEAT
0689:             CH := SYMBUFP^[SYMCURSOR+1];
0690:             SW := SYMBUFP^[SYMCURSOR+2];
0691:             DEL := SYMBUFP^[SYMCURSOR+3];
0692:             CASE CH OF
0693:             'G': GOTOOK := (SW='+');
0694:             'I': IF (SW='+') OR (SW='-') THEN IOCHECK := (SW='+')
0695:                  ELSE STARTINCL;
0696:             'L': LIST := (SW='+');
0697:             'R': RANGECHECK := (SW='+');
0698:             'U': BEGIN SYSCOMP := (SW = '-');
0699:                    RANGECHECK := NOT SYSCOMP;
0700:                    IOCHECK := RANGECHECK;
0701:                    GOTOOK := SYSCOMP
0702:                  END
0703:             END (*CASES*);
0704:             SYMCURSOR := SYMCURSOR+3;
0705:           UNTIL DEL <> ',';
0706:         END;
0707:     SYMCURSOR := SYMCURSOR-1; (* ADJUST *)
0708:    REPEAT
0709:      REPEAT
0710:        SYMCURSOR := SYMCURSOR+1;
0711:        WHILE SYMBUFP^[SYMCURSOR] = CHR(EOL) DO CHECKEND
0712:      UNTIL SYMBUFP^[SYMCURSOR]='*';
0713:    UNTIL (SYMBUFP^[SYMCURSOR+1]=')')
0714:    SYMCURSOR := SYMCURSOR+2;
0715:  END (*COMMENTER*);
0716:
0717:  PROCEDURE STRING;
0718:  VAR
0719:    T: PACKED ARRAY [1..80] OF CHAR;
0720:    TP,NBLANKS,L: INTEGER;
0721:    DUPLE: BOOLEAN;
0722:
0723:  BEGIN
0724:    DUPLE := FALSE; (* INDICATES WHEN '' IS PRESENT *)
0725:    TP := 0; (* INDEX INTO TEMPORARY STRING *)
0726:    REPEAT
```

```
0727:      IF DUPLE THEN SYMCURSOR := SYMCURSOR+1;
0728:      REPEAT
0729:        SYMCURSOR := SYMCURSOR+1;
0730:        TP := TP+1;
0731:        IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
0732:          BEGIN ERROR(202); CHECKEND END;
0733:        T[TP] := SYMBUFP^[SYMCURSOR];
0734:      UNTIL SYMBUFP^[SYMCURSOR]='''';
0735:      DUPLE := TRUE;
0736:    UNTIL SYMBUFP^[SYMCURSOR+1]<>'''';
0737:  1:  TP := TP-1; (* ADJUST *)
0738:    SY := STRINGCONST; OP := NOOP;
0739:    LGTH := TP; (* GROSS *)
0740:    IF TP=1 (* SINGLE CHARACTER CONSTANT *)
0741:      THEN
0742:        VAL.IVAL := ORD(T[1])
0743:      ELSE
0744:        WITH SCONSTDO
0745:          BEGIN
0746:            CCLASS := STRG;
0747:            SLGTH := TP;
0748:            MOVELEFT(T[1],SVAL[1],TP);
0749:            VAL.VALP := SCONST
0750:          END
0751:  END(*STRING*);
0752:
0753:  PROCEDURE NUMBER;
0754:  VAR
0755:    EXPONENT,ENDI,ENDF,ENDE,SIGN,IPART,FPART,EPART,
0756:    ISUM:  INTEGER;
0757:    TIPE: (REALTIPE,INTEGERTIPE);
0758:    RSUM: REAL;
0759:    J: INTEGER;
0760:  BEGIN
0761:    (* TAKES A NUMBER AND DECIDES WHETHER IT'S REAL
0762:       OR INTEGER AND CONVERTS IT TO THE INTERNAL
0763:       FORM. *)
0764:    TIPE := INTEGERTIPE;
0765:    ENDI := 0;
0766:    ENDF := 0;
0767:    ENDE := 0;
0768:    SIGN := 1;
0769:    EPART := 9999; (* OUT OF REACH *)
0770:    IPART := SYMCURSOR; (* INTEGER PART STARTS HERE *)
0771:    REPEAT
0772:      SYMCURSOR := SYMCURSOR+1
0773:    UNTIL (SYMBUFP^[SYMCURSOR]<'0') OR (SYMBUFP^[SYMCURSOR]>'9');
0774:    (* SYMCURSOR NOW POINTS AT FIRST CHARACTER PAST INTEGER PART *)
0775:    ENDI := SYMCURSOR-1; (* MARK THE END OF IPART *)
0776:    IF SYMBUFP^[SYMCURSOR]='.'
0777:      THEN
0778:        IF SYMBUFP^[SYMCURSOR+1]<>'.'  (* WATCH OUT FOR '..' *)
0779:          THEN
0780:            BEGIN
0781:              TIPE := REALTIPE;
0782:              SYMCURSOR := SYMCURSOR+1;
0783:              FPART := SYMCURSOR; (* BEGINNING OF FPART *)
0784:              REPEAT
0785:                SYMCURSOR := SYMCURSOR+1;
0786:              UNTIL (SYMBUFP^[SYMCURSOR]<'0') OR (SYMBUFP^[SYMCURSOR]>'9');
0787:              ENDF := SYMCURSOR-1;
0788:            END;
0789:    IF SYMBUFP^[SYMCURSOR]='E'
0790:      THEN
0791:        BEGIN
0792:          TIPE := REALTIPE;
```

```
0793:          SYMCURSOR := SYMCURSOR+1;
0794:          IF SYMBUFP^[SYMCURSOR]='-'
0795:            THEN
0796:              BEGIN
0797:                SYMCURSOR := SYMCURSOR+1;
0798:                SIGN := -1;
0799:              END
0800:            ELSE
0801:              IF SYMBUFP^[SYMCURSOR]='+'
0802:                THEN
0803:                  SYMCURSOR := SYMCURSOR+1;
0804:          EPART := SYMCURSOR; (* BEGINNING OF EXPONENT *)
0805:          WHILE (SYMBUFP^[SYMCURSOR]>='0') AND (SYMBUFP^[SYMCURSOR]<='9') DO
0806:            SYMCURSOR := SYMCURSOR+1;
0807:          ENDE := SYMCURSOR-1;
0808:          IF ENDE<EPART THEN ERROR(201); (* ERROR IN REAL CONSTANT *)
0809:        END;
0810:    (* NOW CONVERT TO INTERNAL FORM *)
0811:    IF TIPE=INTEGERTIPE
0812:      THEN
0813:        BEGIN
0814:          ISUM := 0;
0815:          FOR J := IPART TO ENDI DO
0816:            BEGIN
0817:              IF (ISUM>3276) OR ((ISUM=3276) AND (SYMBUFP^[J]>'7')) THEN
0818:                  BEGIN ERROR(203); J := ENDI END
0819:              ELSE ISUM := ISUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
0820:            END;
0821:            SY := INTCONST;  OP := NOOP;
0822:            VAL.IVAL := ISUM;
0823:          END
0824:        ELSE
0825:         BEGIN (* REAL NUMBER HERE *)
0826:           RSUM := 0;
0827:           FOR J := IPART TO ENDI DO
0828:             BEGIN
0829:               RSUM := RSUM*10+(ORD(SYMBUFP^[J])-ORD('0'));
0830:             END;
0831:           FOR J := ENDF DOWNTO FPART DO
0832:             RSUM := RSUM+(ORD(SYMBUFP^[J])-ORD('0'))/PWROFTEN(J-FPART+1);
0833:           EXPONENT := 0;
0834:           FOR J := EPART TO ENDE DO
0835:             EXPONENT := EXPONENT*10+ORD(SYMBUFP^[J])-ORD('0');
0836:           IF SIGN=-1
0837:             THEN
0838:               RSUM := RSUM/PWROFTEN(EXPONENT)
0839:             ELSE
0840:               RSUM := RSUM*PWROFTEN(EXPONENT);
0841:           SY := REALCONST;  OP := NOOP;
0842:           NEW(LVP,REEL);
0843:           LVP^.CCLASS := REEL;
0844:           LVP^.RVAL := RSUM;
0845:           VAL.VALP := LVP;
0846:         END;
0847:    SYMCURSOR := SYMCURSOR-1; (* ADJUST FOR POSTERITY *)
0848:  END;
0849:
0850:  BEGIN (* INSYMBOL *)
0851:    OP := NOOP;
0852:  1:  SY := OTHERSY; (* IF NO CASES EXERCISED BLOW UP *)
0853:    CASE SYMBUFP^[SYMCURSOR] OF
0854:    '''':STRING;
0855:    '0','1','2','3','4','5','6','7','8','9':
0856:         NUMBER;
0857:    'A','B','C','D','E','F','G','H','I','J','K','L',
0858:    'M',N','O','P','Q','R','S','T','U','V','W','X',
```

```
0859:    'Y','Z':
0860:         IDSEARCH(SYMCURSOR,SYMBUFP^); (* MAGIC PROC *)
0861:    '': BEGIN COMMENTER(''); GOTO 1 END;
0862:    '(': BEGIN
0863:           IF SYMBUFP^[SYMCURSOR+1]='*' THEN
0864:               BEGIN
0865:                  COMMENTER;
0866:                  GOTO 1; (* GET ANOTHER TOKEN *)
0867:               END
0868:             ELSE
0869:               SY := LPARENT;
0870:         END;
0871:    ')': SY := RPARENT;
0872:    ',': SY := COMMA;
0873:    ' ',' ': BEGIN SYMCURSOR := SYMCURSOR+1; GOTO 1; END;
0874:    '.': BEGIN
0875:           IF SYMBUFP^[SYMCURSOR+1]='.'
0876:             THEN
0877:               BEGIN
0878:                 SYMCURSOR := SYMCURSOR+1;
0879:                 SY := COLON
0880:               END
0881:             ELSE
0882:               SY := PERIOD;
0883:         END;
0884:    ':': IF SYMBUFP^[SYMCURSOR+1]='='
0885:           THEN
0886:             BEGIN
0887:               SYMCURSOR := SYMCURSOR+1;
0888:               SY := BECOMES;
0889:              END
0890:          ELSE
0891:             SY := COLON;
0892:    ';': SY := SEMICOLON;
0893:    ','^':
0894:         SY := ARROW;
0895:    '[': SY := LBRACK;
0896:    ']': SY := RBRACK;
0897:    '*': BEGIN SY := MULOP; OP := MUL END;
0898:    '+': BEGIN SY := ADDOP; OP := PLUS END;
0899:    '-': BEGIN SY := ADDOP; OP := MINUS END;
0900:    '/': BEGIN SY := MULOP; OP := RDIV END;
0901:    '<': BEGIN
0902:           SY := RELOP;
0903:           OP := LTOP;
0904:           CASE SYMBUFP^[SYMCURSOR+1] OF
0905:             '>': BEGIN
0906:                    OP := NEOP;
0907:                    SYMCURSOR := SYMCURSOR+1
0908:                  END;
0909:             '=': BEGIN
0910:                    OP := LEOP;
0911:                    SYMCURSOR := SYMCURSOR+1
0912:                  END
0913:           END;
0914:         END;
0915:    '=': BEGIN SY := RELOP; OP := EQOP END;
0916:    '>': BEGIN
0917:           SY := RELOP;
0918:           IF SYMBUFP^[SYMCURSOR+1]='='
0919:             THEN
0920:               BEGIN
0921:                 OP := GEOP;
0922:                 SYMCURSOR := SYMCURSOR+1;
0923:               END
0924:             ELSE
```

```
0925:               OP := GTOP;
0926:         END
0927:   END (* CASE SYMBUFP^[SYMCURSOR] OF *);
0928:     IF SY=OTHERSY THEN
0929:       IF SYMBUFP^[SYMCURSOR] = CHR(EOL) THEN
0930:         BEGIN CHECKEND; GOTO 1 END
0931:       ELSE ERROR(400);
0932:     SYMCURSOR := SYMCURSOR+1; (* NEXT CALL TALKS ABOUT NEXT TOKEN *)
0933:   END (*INSYMBOL*) ;
0934:
0935:   PROCEDURE ENTERID(FCP: CTP);
0936:     VAR LCP,LCP1: CTP; I: INTEGER;
0937:   BEGIN LCP := DISPLAY[TOP].FNAME;
0938:     IF LCP = NIL THEN DISPLAY[TOP].FNAME := FCP
0939:     ELSE
0940:       BEGIN I := TREESEARCH(LCP,LCP1,FCP^.NAME);
0941:         WHILE I = 0 DO
0942:           BEGIN ERROR(101);
0943:             IF LCP1^.RLINK = NIL THEN I := 1
0944:             ELSE I := TREESEARCH(LCP1^.RLINK,LCP1,FCP^.NAME)
0945:           END;
0946:         IF I = 1 THEN LCP1^.RLINK := FCP ELSE LCP1^.LLINK := FCP
0947:       END;
0948:     FCP^.LLINK := NIL; FCP^.RLINK := NIL
0949:   END (*ENTERID*) ;
0950:
0951:     PROCEDURE SEARCHSECTION(FCP: CTP; VAR FCP1: CTP);
0952:     BEGIN
0953:       IF FCP <> NIL THEN
0954:         IF TREESEARCH(FCP,FCP1,ID) = 0 THEN (*NADA*)
0955:         ELSE FCP1 := NIL
0956:       ELSE FCP1 := NIL
0957:     END (*SEARCHSECTION*) ;
0958:
0959:     PROCEDURE SEARCHID(FIDCLS: SETOFIDS; VAR FCP: CTP);
0960:       LABEL 1; VAR LCP: CTP;
0961:     BEGIN
0962:       FOR DISX := TOP DOWNTO 0 DO
0963:         BEGIN LCP := DISPLAY[DISX].FNAME;
0964:           IF LCP <> NIL THEN
0965:             IF TREESEARCH(LCP,LCP,ID) = 0 THEN
0966:               IF LCP^.KLASS IN FIDCLS THEN GOTO 1
0967:               ELSE
0968:                 IF PRTERR THEN ERROR(103)
0969:                 ELSE LCP := NIL
0970:             ELSE LCP := NIL
0971:         END;
0972:       IF PRTERR THEN
0973:         BEGIN ERROR(104);
0974:           IF TYPES IN FIDCLS THEN LCP := UTYPPTR
0975:           ELSE
0976:             IF VARS IN FIDCLS THEN LCP := UVARPTR
0977:             ELSE
0978:               IF FIELD IN FIDCLS THEN LCP := UFLDPTR
0979:               ELSE
0980:                 IF KONST IN FIDCLS THEN LCP := UCSTPTR
0981:                 ELSE
0982:                   IF PROC IN FIDCLS THEN LCP := UPRCPTR
0983:                   ELSE LCP := UFCTPTR
0984:         END;
0985:   1:  FCP := LCP
0986:     END (*SEARCHID*) ;
0987:
0988:     PROCEDURE GETBOUNDS(FSP: STP; VAR FMIN,FMAX: INTEGER);
0989:     BEGIN
0990:       WITH FSP^ DO
```

```
0991:        IF FORM = SUBRANGE THEN
0992:          BEGIN FMIN := MIN.IVAL; FMAX := MAX.IVAL END
0993:        ELSE
0994:          BEGIN FMIN := 0;
0995:            IF FSP = CHARPTR THEN FMAX := 255
0996:            ELSE
0997:              IF FSP^.FCONST <> NIL THEN
0998:                FMAX := FSP^.FCONST^.VALUES.IVAL
0999:              ELSE FMAX := 0
1000:          END
1001:    END (*GETBOUNDS*) ;
1002:
1003:    PROCEDURE SKIP(FSYS: SETOFSYS);
1004:    BEGIN WHILE NOT(SY IN FSYS) DO INSYMBOL
1005:    END (*SKIP*) ;
1006:
1007:    FUNCTION PAOFCHAR(FSP: STP): BOOLEAN;
1008:    BEGIN PAOFCHAR := FALSE;
1009:      IF FSP <> NIL THEN
1010:        IF FSP^.FORM = ARRAYS THEN
1011:          PAOFCHAR := FSP^.AISPACKD AND (FSP^.AELTYPE = CHARPTR)
1012:    END (*PAOFCHAR*) ;
1013:
1014:    FUNCTION STRGTYPE(FSP: STP) : BOOLEAN;
1015:    BEGIN STRGTYPE := FALSE;
1016:      IF PAOFCHAR(FSP) THEN STRGTYPE := FSP^.AISSTRNG
1017:    END (*STRGTYPE*) ;
1018:
1019:    PROCEDURE CONSTANT(FSYS: SETOFSYS; VAR FSP: STP; VAR FVALU: VALU);
1020:      VAR LSP: STP; LCP: CTP; SIGN: (NONE,POS,NEG);
1021:          LVP: CSP;
1022:    BEGIN LSP := NIL; FVALU.IVAL := 0;
1023:      IF NOT(SY IN CONSTBEGSYS) THEN
1024:        BEGIN ERROR(50); SKIP(FSYS+CONSTBEGSYS) END;
1025:      IF SY IN CONSTBEGSYS THEN
1026:        BEGIN
1027:          IF SY = STRINGCONSTSY THEN
1028:            BEGIN
1029:              IF LGTH = 1 THEN LSP := CHARPTR
1030:              ELSE
1031:                BEGIN
1032:                  NEW(LSP,ARRAYS,TRUE,TRUE);
1033:                  LSP^ := STRGPTR^;
1034:                  LSP^.MAXLENG := LGTH;
1035:                  LSP^.INXTYPE := NIL;
1036:                  NEW(LVP);
1037:                  LVP^ := VAL.VALP^;
1038:                  VAL.VALP := LVP
1039:                END;
1040:              FVALU := VAL; INSYMBOL
1041:            END
1042:          ELSE
1043:            BEGIN
1044:              SIGN := NONE;
1045:              IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
1046:                BEGIN IF OP = PLUS THEN SIGN := POS ELSE SIGN := NEG;
1047:                  INSYMBOL
1048:                END;
1049:              IF SY = IDENT THEN
1050:                BEGIN SEARCHID([KONST],LCP);
1051:                  WITH LCP^ DO
1052:                    BEGIN LSP := IDTYPE; FVALU := VALUES END;
1053:                  IF SIGN <> NONE THEN
1054:                    IF LSP = INTPTR THEN
1055:                      BEGIN IF SIGN = NEG THEN
1056:                        FVALU.IVAL := -FVALU.IVAL END
```

```
1057:                   ELSE
1058:                     IF LSP = REALPTR THEN
1059:                       BEGIN
1060:                         IF SIGN = NEG THEN
1061:                           BEGIN NEW(LVP,REEL);
1062:                             LVP^.CCLASS := REEL;
1063:                             LVP^.RVAL := -FVALU.VALP^.RVAL;
1064:                             FVALU.VALP := LVP;
1065:                           END
1066:                         END
1067:                       ELSE ERROR(105);
1068:                 INSYMBOL;
1069:               END
1070:             ELSE
1071:               IF SY = INTCONST THEN
1072:                 BEGIN IF SIGN = NEG THEN VAL.IVAL := -VAL.IVAL;
1073:                   LSP := INTPTR; FVALU := VAL; INSYMBOL
1074:                 END
1075:               ELSE
1076:                 IF SY = REALCONST THEN
1077:                   BEGIN IF SIGN = NEG THEN
1078:                           VAL.VALP^.RVAL := -VAL.VALP^.RVAL;
1079:                     LSP := REALPTR; FVALU := VAL; INSYMBOL
1080:                   END
1081:                 ELSE
1082:                   BEGIN ERROR(106); SKIP(FSYS) END
1083:           END;
1084:         IF NOT (SY IN FSYS) THEN
1085:           BEGIN ERROR(6); SKIP(FSYS) END
1086:         END;
1087:     FSP := LSP
1088:   END (*CONSTANT*) ;
1089:
1090:   FUNCTION COMPTYPES(FSP1,FSP2: STP) : BOOLEAN;
1091:     VAR NXT1,NXT2: CTP; COMP: BOOLEAN;
1092:       LTESTP1,LTESTP2 : TESTP;
1093:   BEGIN
1094:     IF FSP1 = FSP2 THEN COMPTYPES := TRUE
1095:     ELSE
1096:       IF (FSP1 = NIL) OR (FSP2 = NIL) THEN COMPTYPES := TRUE
1097:       ELSE
1098:         IF FSP1^.FORM = FSP2^.FORM THEN
1099:           CASE FSP1^.FORM OF
1100:             SCALAR:
1101:               COMPTYPES := FALSE;
1102:             SUBRANGE:
1103:               COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,
1104:                                      FSP2^.RANGETYPE);
1105:             POINTER:
1106:                 BEGIN
1107:                   COMP := FALSE; LTESTP1 := GLOBTESTP;
1108:                   LTESTP2 := GLOBTESTP;
1109:                   WHILE LTESTP1 <> NIL DO
1110:                     WITH LTESTP1^ DO
1111:                       BEGIN
1112:                         IF (ELT1 = FSP1^.ELTYPE) AND
1113:                           (ELT2 = FSP2^.ELTYPE) THEN COMP := TRUE;
1114:                         LTESTP1 := LASTTESTP
1115:                       END;
1116:                   IF NOT COMP THEN
1117:                     BEGIN NEW(LTESTP1);
1118:                       WITH LTESTP1^ DO
1119:                         BEGIN ELT1 := FSP1^.ELTYPE;
1120:                           ELT2 := FSP2^.ELTYPE;
1121:                           LASTTESTP := GLOBTESTP
1122:                         END;
```

```
1123:                       GLOBTESTP := LTESTP1;
1124:                       COMP := COMPTYPES(FSP1^.ELTYPE,FSP2^.ELTYPE)
1125:                     END;
1126:                   COMPTYPES := COMP; GLOBTESTP := LTESTP2
1127:                 END;
1128:               POWER:
1129:                 COMPTYPES := COMPTYPES(FSP1^.ELSET,FSP2^.ELSET);
1130:               ARRAYS:
1131:                 BEGIN
1132:                   COMP := COMPTYPES(FSP1^.AELTYPE,FSP2^.AELTYPE)
1133:                           AND (FSP1^.AISPACKD = FSP2^.AISPACKD);
1134:                   IF COMP AND FSP1^.AISPACKD THEN
1135:                       COMP := (FSP1^.ELSPERWD = FSP2^.ELSPERWD)
1136:                               AND (FSP1^.ELWIDTH = FSP2^.ELWIDTH)
1137:                               AND (FSP1^.AISSTRNG = FSP2^.AISSTRNG);
1138:                   IF COMP AND NOT STRGTYPE(FSP1) THEN
1139:                     COMP := (FSP1^.SIZE = FSP2^.SIZE);
1140:                   COMPTYPES := COMP;
1141:                 END;
1142:               RECORDS:
1143:                 BEGIN NXT1 := FSP1^.FSTFLD; NXT2 := FSP2^.FSTFLD;
1144:                   COMP := TRUE;
1145:                   WHILE (NXT1 <> NIL) AND (NXT2 <> NIL) AND COMP DO
1146:                     BEGIN COMP:=COMPTYPES(NXT1^.IDTYPE,NXT2^.IDTYPE);
1147:                       NXT1 := NXT1^.NEXT; NXT2 := NXT2^.NEXT
1148:                     END;
1149:                   COMPTYPES := COMP AND (NXT1 = NIL) AND (NXT2 = NIL)
1150:                               AND (FSP1^.RECVAR = NIL)
1151:                               AND (FSP2^.RECVAR = NIL)
1152:                 END;
1153:               FILES:
1154:                 COMPTYPES := COMPTYPES(FSP1^.FILTYPE,FSP2^.FILTYPE)
1155:             END (*CASE*)
1156:           ELSE (*FSP1^.FORM <> FSP2^.FORM*)
1157:             IF FSP1^.FORM = SUBRANGE THEN
1158:               COMPTYPES := COMPTYPES(FSP1^.RANGETYPE,FSP2)
1159:             ELSE
1160:               IF FSP2^.FORM = SUBRANGE THEN
1161:                 COMPTYPES := COMPTYPES(FSP1,FSP2^.RANGETYPE)
1162:               ELSE COMPTYPES := FALSE
1163:     END (*COMPTYPES*) ;
1164:
1165:   (*$I XCOMP:C.TEXT *)
1166:
1167:     PROCEDURE TYP(FSYS: SETOFSYS; VAR FSP: STP; VAR FSIZE: ADDRRANGE);
1168:       VAR LSP,LSP1,LSP2: STP; OLDTOP: DISPRANGE; LCP: CTP;
1169:           LSIZE,DISPL: ADDRRANGE; LMIN,LMAX: INTEGER;
1170:           PACKING: BOOLEAN; NEXTBIT,NUMBITS: BITRANGE;
1171:
1172:       PROCEDURE SIMPLETYPE(FSYS:SETOFSYS; VAR FSP:STP; VAR FSIZE:ADDRRANGE);
1173:         VAR LSP,LSP1: STP; LCP,LCP1: CTP; TTOP: DISPRANGE;
1174:             LCNT: INTEGER; LVALU: VALU;
1175:       BEGIN FSIZE := 1;
1176:         IF NOT (SY IN SIMPTYPEBEGSYS) THEN
1177:           BEGIN ERROR(1); SKIP(FSYS + SIMPTYPEBEGSYS) END;
1178:         IF SY IN SIMPTYPEBEGSYS THEN
1179:           BEGIN
1180:             IF SY = LPARENT THEN
1181:               BEGIN TTOP := TOP;
1182:                 WHILE DISPLAY[TOP].OCCUR <> BLCK DO TOP := TOP - 1;
1183:                 NEW(LSP,SCALAR,DECLARED);
1184:                 WITH LSP^ DO
1185:                   BEGIN SIZE := INTSIZE; FORM := SCALAR;
1186:                     SCALKIND := DECLARED
1187:                   END;
1188:                 LCP1 := NIL; LCNT := 0;
```

```
1189:               REPEAT INSYMBOL;
1190:                 IF SY = IDENT THEN
1191:                   BEGIN NEW(LCP,KONST);
1192:                     WITH LCP^ DO
1193:                       BEGIN NAME := ID; IDTYPE := LSP; NEXT := LCP1;
1194:                         VALUES.IVAL := LCNT; KLASS := KONST
1195:                       END;
1196:                     ENTERID(LCP);
1197:                     LCNT := LCNT + 1;
1198:                     LCP1 := LCP; INSYMBOL
1199:                   END
1200:                 ELSE ERROR(2);
1201:                 IF NOT (SY IN FSYS + [COMMA,RPARENT]) THEN
1202:                   BEGIN ERROR(6); SKIP(FSYS + [COMMA,RPARENT]) END
1203:               UNTIL SY <> COMMA;
1204:               LSP^.FCONST := LCP1; TOP := TTOP;
1205:               IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
1206:             END
1207:           ELSE
1208:             BEGIN
1209:               IF SY = IDENT THEN
1210:                 BEGIN SEARCHID([TYPES,KONST],LCP);
1211:                   INSYMBOL;
1212:                   IF LCP^.KLASS = KONST THEN
1213:                     BEGIN NEW(LSP,SUBRANGE);
1214:                       WITH LSP^, LCP^ DO
1215:                         BEGIN RANGETYPE := IDTYPE; FORM := SUBRANGE;
1216:                           IF STRGTYPE(RANGETYPE) THEN
1217:                             BEGIN ERROR(148); RANGETYPE := NIL END;
1218:                           MIN := VALUES; SIZE := INTSIZE
1219:                         END;
1220:                       IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1221:                       CONSTANT(FSYS,LSP1,LVALU);
1222:                       LSP^.MAX := LVALU;
1223:                       IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
1224:                     END
1225:                   ELSE
1226:                     BEGIN LSP := LCP^.IDTYPE;
1227:                       IF (LSP = STRGPTR) AND (SY = LBRACK) THEN
1228:                         BEGIN INSYMBOL;
1229:                           CONSTANT(FSYS + [RBRACK],LSP1,LVALU);
1230:                           IF LSP1 = INTPTR THEN
1231:                             BEGIN
1232:                               IF (LVALU.IVAL <= 0) OR
1233:                                  (LVALU.IVAL > STRGLGTH) THEN
1234:                                 BEGIN ERROR(203);
1235:                                   LVALU.IVAL := DEFSTRGLGTH
1236:                                 END;
1237:                               IF LVALU.IVAL <> DEFSTRGLGTH THEN
1238:                                 BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
1239:                                   LSP^ := STRGPTR^;
1240:                                   WITH LSP^,LVALU DO
1241:                                     BEGIN MAXLENG := IVAL;
1242:                                       SIZE := (IVAL+CHRSPERWD) DIV CHRSPERWD
1243:                                     END
1244:                                 END
1245:                             END
1246:                           ELSE ERROR(15);
1247:                           IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
1248:                         END;
1249:                       IF LSP <> NIL THEN FSIZE := LSP^.SIZE
1250:                     END
1251:                 END (*SY = IDENT*)
1252:               ELSE
1253:                 BEGIN NEW(LSP,SUBRANGE); LSP^.FORM := SUBRANGE;
1254:                   CONSTANT(FSYS + [COLON],LSP1,LVALU);
```

```
1255:                   IF STRGTYPE(LSP1) THEN
1256:                     BEGIN ERROR(148); LSP1 := NIL END;
1257:                   WITH LSP^ DO
1258:                     BEGIN RANGETYPE:=LSP1; MIN:=LVALU; SIZE:=INTSIZE END;
1259:                   IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1260:                   CONSTANT(FSYS,LSP1,LVALU);
1261:                   LSP^.MAX := LVALU;
1262:                   IF LSP^.RANGETYPE <> LSP1 THEN ERROR(107)
1263:                 END;
1264:               IF LSP <> NIL THEN
1265:                 WITH LSP^ DO
1266:                   IF FORM = SUBRANGE THEN
1267:                     IF RANGETYPE <> NIL THEN
1268:                       IF RANGETYPE = REALPTR THEN ERROR(399)
1269:                       ELSE
1270:                         IF MIN.IVAL > MAX.IVAL THEN
1271:                           BEGIN ERROR(102); MAX.IVAL := MIN.IVAL END
1272:             END;
1273:           FSP := LSP;
1274:           IF NOT (SY IN FSYS) THEN
1275:             BEGIN ERROR(6); SKIP(FSYS) END
1276:         END
1277:           ELSE FSP := NIL
1278:     END (*SIMPLETYPE*) ;
1279:
1280:     FUNCTION PACKABLE(FSP: STP): BOOLEAN;
1281:       VAR LMIN,LMAX: INTEGER;
1282:     BEGIN PACKABLE := FALSE;
1283:       IF (FSP <> NIL) AND PACKING THEN
1284:         WITH FSP^ DO
1285:           CASE FORM OF
1286:             SUBRANGE,
1287:             SCALAR:  IF (FSP <> INTPTR) AND (FSP <> REALPTR) THEN
1288:                        BEGIN GETBOUNDS(FSP,LMIN,LMAX);
1289:                          IF LMIN >= 0 THEN
1290:                            BEGIN PACKABLE := TRUE;
1291:                              NUMBITS := 1; LMIN := 1;
1292:                              WHILE LMIN < LMAX DO
1293:                                BEGIN LMIN := LMIN + 1;
1294:                                  LMIN := LMIN + LMIN - 1;
1295:                                  NUMBITS := NUMBITS + 1
1296:                                END
1297:                            END
1298:                        END;
1299:             POWER:   IF PACKABLE(ELSET) THEN
1300:                        BEGIN GETBOUNDS(ELSET,LMIN,LMAX);
1301:                          LMAX := LMAX + 1;
1302:                          IF LMAX < BITSPERWD THEN
1303:                            BEGIN PACKABLE := TRUE;
1304:                              NUMBITS := LMAX
1305:                            END
1306:                        END
1307:           END (* CASES *);
1308:     END (*PACKABLE*) ;
1309:
1310:     PROCEDURE FIELDLIST(FSYS: SETOFSYS; VAR FRECVAR: STP);
1311:       VAR LCP,LCP1,NXT,NXT1,LAST: CTP; LSP,LSP1,LSP2,LSP3,LSP4: STP;
1312:           MINSIZE,MAXSIZE,LSIZE: ADDRRANGE; LVALU: VALU;
1313:           MAXBIT,MINBIT: BITRANGE;
1314:
1315:       PROCEDURE ALLOCATE(FCP: CTP);
1316:         VAR ONBOUND: BOOLEAN;
1317:       BEGIN ONBOUND := FALSE;
1318:         WITH FCP^ DO
1319:           IF PACKABLE(IDTYPE) THEN
1320:             BEGIN
```

```
1321:               IF (NUMBITS + NEXTBIT) > BITSPERWD THEN
1322:                 BEGIN DISPL := DISPL + 1; NEXTBIT := 0; ONBOUND := TRUE END;
1323:               FLDADDR := DISPL; FISPACKD := TRUE;
1324:               FLDWIDTH := NUMBITS; FLDRBIT := NEXTBIT;
1325:               NEXTBIT := NEXTBIT + NUMBITS
1326:             END
1327:           ELSE
1328:             BEGIN DISPL := DISPL + ORD(NEXTBIT > 0);
1329:               NEXTBIT := 0; ONBOUND := TRUE;
1330:               FISPACKD := FALSE; FLDADDR := DISPL;
1331:               IF IDTYPE <> NIL THEN
1332:                 DISPL := DISPL + IDTYPE^.SIZE
1333:             END;
1334:         IF ONBOUND AND (LAST <> NIL) THEN
1335:           WITH LAST^ DO
1336:             IF FISPACKD THEN
1337:               IF FLDRBIT = 0 THEN FISPACKD := FALSE
1338:               ELSE
1339:                 IF (FLDWIDTH <= 8) AND (FLDRBIT <= 8) THEN
1340:                   BEGIN FLDWIDTH := 8; FLDRBIT := 8 END
1341:       END (*ALLOCATE*) ;
1342:
1343:       PROCEDURE VARIANTLIST;
1344:         VAR GOTTAGNAME: BOOLEAN;
1345:       BEGIN NEW(LSP,TAGFLD);
1346:         WITH LSP^ DO
1347:           BEGIN TAGFIELDP := NIL; FSTVAR := NIL; FORM := TAGFLD END;
1348:         FRECVAR := LSP;
1349:         INSYMBOL;
1350:         IF SY = IDENT THEN
1351:           BEGIN
1352:             IF PACKING THEN NEW(LCP,FIELD,TRUE)
1353:             ELSE NEW(LCP,FIELD,FALSE);
1354:             WITH LCP^ DO
1355:               BEGIN IDTYPE := NIL; KLASS:=FIELD;
1356:                 NEXT := NIL; FISPACKD := FALSE
1357:               END;
1358:             GOTTAGNAME := FALSE; PRTERR := FALSE;
1359:             SEARCHID([TYPES],LCP1); PRTERR := TRUE;
1360:             IF LCP1 = NIL THEN
1361:               BEGIN GOTTAGNAME := TRUE;
1362:                 LCP^.NAME := ID; ENTERID(LCP); INSYMBOL;
1363:                 IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
1364:               END;
1365:             IF SY = IDENT THEN
1366:               BEGIN SEARCHID([TYPES],LCP1);
1367:                 LSP1 := LCP1^.IDTYPE;
1368:                 IF LSP1 <> NIL THEN
1369:                   BEGIN
1370:                     IF LSP1^.FORM <= SUBRANGE THEN
1371:                       BEGIN
1372:                         IF COMPTYPES(REALPTR,LSP1) THEN ERROR(109);
1373:                         LCP^.IDTYPE := LSP1; LSP^.TAGFIELDP := LCP;
1374:                         IF GOTTAGNAME THEN ALLOCATE(LCP)
1375:                       END
1376:                     ELSE ERROR(110)
1377:                   END;
1378:                 INSYMBOL
1379:               END
1380:             ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END
1381:           END
1382:         ELSE BEGIN ERROR(2); SKIP(FSYS + [OFSY,LPARENT]) END;
1383:         LSP^.SIZE := DISPL + ORD(NEXTBIT > 0);
1384:         IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1385:         LSP1 := NIL; MINSIZE := DISPL; MAXSIZE := DISPL;
1386:         MINBIT := NEXTBIT; MAXBIT := NEXTBIT;
```

```
1387:          REPEAT LSP2 := NIL;
1388:            REPEAT CONSTANT(FSYS + [COMMA,COLON,LPARENT],LSP3,LVALU);
1389:              IF LSP^.TAGFIELDP <> NIL THEN
1390:                IF NOT COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP3) THEN
1391:                  ERROR(111);
1392:              NEW(LSP3,VARIANT);
1393:              WITH LSP3^ DO
1394:                BEGIN NXTVAR := LSP1; SUBVAR := LSP2;
1395:                  VARVAL := LVALU; FORM := VARIANT
1396:                END;
1397:              LSP1 := LSP3; LSP2 := LSP3;
1398:              TEST := SY <> COMMA;
1399:              IF NOT TEST THEN INSYMBOL
1400:            UNTIL TEST;
1401:            IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1402:            IF SY = LPARENT THEN INSYMBOL ELSE ERROR(9);
1403:            IF SY = RPARENT THEN LSP2 := NIL
1404:            ELSE
1405:              FIELDLIST(FSYS + [RPARENT,SEMICOLON],LSP2);
1406:            IF DISPL > MAXSIZE THEN
1407:              BEGIN MAXSIZE := DISPL; MAXBIT := NEXTBIT END
1408:            ELSE
1409:              IF (DISPL = MAXSIZE) AND (NEXTBIT > MAXBIT) THEN
1410:                MAXBIT := NEXTBIT;
1411:            WHILE LSP3 <> NIL DO
1412:              BEGIN LSP4 := LSP3^.SUBVAR; LSP3^.SUBVAR := LSP2;
1413:                LSP3^.SIZE := DISPL + ORD(NEXTBIT > 0);
1414:                LSP3 := LSP4
1415:              END;
1416:            IF SY = RPARENT THEN
1417:              BEGIN INSYMBOL;
1418:                IF NOT (SY IN FSYS + [SEMICOLON]) THEN
1419:                  BEGIN ERROR(6); SKIP(FSYS + [SEMICOLON]) END
1420:              END
1421:            ELSE ERROR(4);
1422:            TEST := SY <> SEMICOLON;
1423:            IF NOT TEST THEN
1424:              BEGIN INSYMBOL;
1425:                DISPL := MINSIZE; NEXTBIT := MINBIT
1426:              END
1427:          UNTIL TEST;
1428:          DISPL := MAXSIZE; NEXTBIT := MAXBIT;
1429:          LSP^.FSTVAR := LSP1
1430:        END (*VARIANTLIST*) ;
1431:
1432:      BEGIN (*FIELDLIST*)
1433:        NXT1 := NIL; LSP := NIL; LAST := NIL;
1434:        IF NOT (SY IN [IDENT,CASESY]) THEN
1435:          BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END;
1436:        WHILE SY = IDENT DO
1437:          BEGIN NXT := NXT1;
1438:            REPEAT
1439:              IF SY = IDENT THEN
1440:                BEGIN
1441:                  IF PACKING THEN NEW(LCP,FIELD,TRUE)
1442:                  ELSE NEW(LCP,FIELD,FALSE);
1443:                  WITH LCP^ DO
1444:                    BEGIN NAME := ID; IDTYPE := NIL; NEXT := NXT;
1445:                      KLASS := FIELD; FISPACKD := FALSE
1446:                    END;
1447:                  NXT := LCP;
1448:                  ENTERID(LCP);
1449:                  INSYMBOL
1450:                END
1451:              ELSE ERROR(2);
1452:              IF NOT (SY IN [COMMA,COLON]) THEN
```

```
1453:               BEGIN ERROR(6); SKIP(FSYS + [COMMA,COLON,SEMICOLON,CASESY]) END;
1454:             TEST := SY <> COMMA;
1455:             IF NOT TEST  THEN INSYMBOL
1456:           UNTIL TEST;
1457:           IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
1458:           TYP(FSYS + [CASESY,SEMICOLON],LSP,LSIZE);
1459:           IF LSP <> NIL THEN
1460:             IF LSP^.FORM = FILES THEN ERROR(108);
1461:           WHILE NXT <> NXT1 DO
1462:             WITH NXT^ DO
1463:               BEGIN IDTYPE := LSP; ALLOCATE(NXT);
1464:                 IF NEXT = NXT1 THEN LAST := NXT;
1465:                 NXT := NEXT
1466:               END;
1467:           NXT1 := LCP;
1468:           IF SY = SEMICOLON THEN
1469:             BEGIN INSYMBOL;
1470:               IF NOT (SY IN [IDENT,CASESY]) THEN
1471:                 BEGIN ERROR(19); SKIP(FSYS + [IDENT,CASESY]) END
1472:             END
1473:         END (*WHILE*);
1474:       NXT := NIL;
1475:       WHILE NXT1 <> NIL DO
1476:         WITH NXT1^ DO
1477:           BEGIN LCP := NEXT; NEXT := NXT; NXT := NXT1; NXT1 := LCP END;
1478:       IF SY = CASESY THEN VARIANTLIST
1479:       ELSE FRECVAR := NIL
1480:     END (*FIELDLIST*) ;
1481:
1482:     PROCEDURE POINTERTYPE;
1483:     BEGIN NEW(LSP,POINTER); FSP := LSP;
1484:       WITH LSP^ DO
1485:         BEGIN ELTYPE := NIL; SIZE := PTRSIZE; FORM := POINTER END;
1486:       INSYMBOL;
1487:       IF SY = IDENT THEN
1488:         BEGIN PRTERR := FALSE;
1489:           SEARCHID([TYPES],LCP); PRTERR := TRUE;
1490:           IF LCP = NIL THEN   (*FORWARD REFERENCED TYPE ID*)
1491:             BEGIN NEW(LCP,TYPES);
1492:               WITH LCP^ DO
1493:                 BEGIN NAME := ID; IDTYPE := LSP;
1494:                   NEXT := FWPTR; KLASS := TYPES
1495:                 END;
1496:               FWPTR := LCP
1497:             END
1498:           ELSE
1499:             BEGIN
1500:               IF LCP^.IDTYPE <> NIL THEN
1501:                 IF (LCP^.IDTYPE^.FORM <> FILES) OR SYSCOMP THEN
1502:                   LSP^.ELTYPE := LCP^.IDTYPE
1503:                 ELSE ERROR(108)
1504:             END;
1505:           INSYMBOL;
1506:         END
1507:       ELSE ERROR(2)
1508:     END (*POINTERTYPE*) ;
1509:
1510:   BEGIN (*TYP*)
1511:     PACKING := FALSE;
1512:     IF NOT (SY IN TYPEBEGSYS) THEN
1513:        BEGIN ERROR(10); SKIP(FSYS + TYPEBEGSYS) END;
1514:     IF SY IN TYPEBEGSYS THEN
1515:       BEGIN
1516:         IF SY IN SIMPTYPEBEGSYS THEN SIMPLETYPE(FSYS,FSP,FSIZE)
1517:         ELSE
1518:   (*^*)   IF SY = ARROW THEN POINTERTYPE
```

```
1519:           ELSE
1520:             BEGIN
1521:               IF SY = PACKEDSY THEN
1522:                 BEGIN INSYMBOL; PACKING := TRUE;
1523:                   IF NOT (SY IN TYPEDELS) THEN
1524:                     BEGIN ERROR(10); SKIP(FSYS + TYPEDELS) END
1525:                 END;
1526:   (*ARRAY*)   IF SY = ARRAYSY THEN
1527:                 BEGIN INSYMBOL;
1528:                   IF SY = LBRACK THEN INSYMBOL ELSE ERROR(11);
1529:                   LSP1 := NIL;
1530:                   REPEAT
1531:                     IF PACKING THEN NEW(LSP,ARRAYS,TRUE,FALSE)
1532:                     ELSE NEW(LSP,ARRAYS,FALSE);
1533:                     WITH LSP^ DO
1534:                       BEGIN AELTYPE := LSP1; INXTYPE := NIL;
1535:                         IF PACKING THEN AISSTRNG := FALSE;
1536:                         AISPACKD := FALSE;  FORM := ARRAYS
1537:                       END;
1538:                     LSP1 := LSP;
1539:                     SIMPLETYPE(FSYS + [COMMA,RBRACK,OFSY],LSP2,LSIZE);
1540:                     LSP1^.SIZE := LSIZE;
1541:                     IF LSP2 <> NIL THEN
1542:                       IF LSP2^.FORM <= SUBRANGE THEN
1543:                         BEGIN
1544:                           IF LSP2 = REALPTR THEN
1545:                             BEGIN ERROR(109); LSP2 := NIL END
1546:                           ELSE
1547:                             IF LSP2 = INTPTR THEN
1548:                               BEGIN ERROR(149); LSP2 := NIL END;
1549:                           LSP^.INXTYPE := LSP2
1550:                         END
1551:                       ELSE BEGIN ERROR(113); LSP2 := NIL END;
1552:                     TEST := SY <> COMMA;
1553:                     IF NOT TEST THEN INSYMBOL
1554:                   UNTIL TEST;
1555:                   IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12);
1556:                   IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1557:                   TYP(FSYS,LSP,LSIZE);
1558:                   IF LSP <> NIL THEN
1559:                     IF LSP^.FORM = FILES THEN ERROR(108);
1560:                   IF PACKABLE(LSP) THEN
1561:                     IF NUMBITS + NUMBITS <= BITSPERWD THEN
1562:                       WITH LSP1^ DO
1563:                         BEGIN AISPACKD := TRUE;
1564:                           ELSPERWD := BITSPERWD DIV NUMBITS;
1565:                           ELWIDTH := NUMBITS
1566:                         END;
1567:                   REPEAT
1568:                     WITH LSP1^ DO
1569:                       BEGIN LSP2 := AELTYPE; AELTYPE := LSP;
1570:                         IF INXTYPE <> NIL THEN
1571:                           BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
1572:                             IF AISPACKD THEN
1573:                               LSIZE := (LMAX-LMIN+ELSPERWD)
1574:                                                   DIV ELSPERWD
1575:                             ELSE
1576:                               LSIZE := LSIZE*(LMAX - LMIN + 1);
1577:                             IF LSIZE <= 0 THEN
1578:                               BEGIN ERROR(398); LSIZE := 1 END;
1579:                             SIZE := LSIZE
1580:                           END
1581:                       END;
1582:                     LSP := LSP1; LSP1 := LSP2
1583:                   UNTIL LSP1 = NIL
1584:                 END
```

```
1585:                ELSE
1586:    (*RECORD*)    IF SY = RECORDSY THEN
1587:                    BEGIN INSYMBOL;
1588:                      OLDTOP := TOP;
1589:                      IF TOP < DISPLIMIT THEN
1590:                        BEGIN TOP := TOP + 1;
1591:                          WITH DISPLAY[TOP] DO
1592:                            BEGIN FNAME := NIL; OCCUR := REC END
1593:                        END
1594:                      ELSE ERROR(250);
1595:                      DISPL := 0; NEXTBIT := 0;
1596:                      FIELDLIST(FSYS-[SEMICOLON]+[ENDSY],LSP1);
1597:                      DISPL := DISPL + ORD(NEXTBIT > 0);
1598:                      NEW(LSP,RECORDS);
1599:                      WITH LSP^ DO
1600:                        BEGIN FSTFLD := DISPLAY[TOP].FNAME;
1601:                          RECVAR := LSP1; SIZE := DISPL;
1602:                          FORM := RECORDS
1603:                        END;
1604:                      TOP := OLDTOP;
1605:                      IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
1606:                    END
1607:                  ELSE
1608:    (*SET*)         IF SY = SETSY THEN
1609:                      BEGIN INSYMBOL;
1610:                        IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
1611:                        SIMPLETYPE(FSYS,LSP1,LSIZE);
1612:                        IF LSP1 <> NIL THEN
1613:                          IF LSP1^.FORM > SUBRANGE
1614:                            BEGIN ERROR(115); LSP1 := NIL END
1615:                          ELSE
1616:                            IF LSP1 = REALPTR THEN
1617:                              BEGIN ERROR(114); LSP1 := NIL END;
1618:                        NEW(LSP,POWER);
1619:                        WITH LSP^ DO
1620:                          BEGIN ELSET := LSP1; FORM := POWER;
1621:                            IF LSP1 <> NIL THEN
1622:                              BEGIN GETBOUNDS(LSP1,LMIN,LMAX);
1623:                                SIZE := (LMAX + BITSPERWD) DIV BITSPERWD;
1624:                              END
1625:                            ELSE SIZE := 0
1626:                          END
1627:                      END
1628:                    ELSE
1629:    (*FILE*)          IF SY = FILESY THEN
1630:                        BEGIN INSYMBOL; NEW(LSP,FILES);
1631:                          WITH LSP^ DO
1632:                            BEGIN FORM := FILES; FILTYPE := NIL END;
1633:                          IF SY = OFSY THEN
1634:                            BEGIN INSYMBOL; TYP(FSYS,LSP1,LSIZE) END
1635:                          ELSE LSP1 := NIL;
1636:                          LSP^.FILTYPE := LSP1;
1637:                          IF LSP1 <> NIL THEN
1638:                            LSP^.SIZE := FILESIZE + LSP1^.SIZE
1639:                          ELSE LSP^.SIZE := NILFILESIZE
1640:                        END;
1641:                FSP := LSP
1642:              END;
1643:          IF NOT (SY IN FSYS) THEN
1644:            BEGIN ERROR(6); SKIP(FSYS) END
1645:        END
1646:      ELSE FSP := NIL;
1647:      IF FSP = NIL THEN FSIZE := 1 ELSE FSIZE := FSP^.SIZE
1648:    END (*TYP*) ;
1649:
1650:    PROCEDURE GENLDC(IVAL: INTEGER); FORWARD;
```

```
1651:
1652:    PROCEDURE GENBYTE(FBYTE: INTEGER);
1653:    BEGIN
1654:      CODEP^[IC] := CHR(FBYTE); IC := IC+1
1655:    END (*GENBYTE*) ;
1656:
1657:    PROCEDURE GENWORD(FWORD: INTEGER);
1658:    BEGIN
1659:      IF ODD(IC) THEN IC := IC + 1;
1660:      MOVELEFT(FWORD,CODEP^[IC],2);
1661:      IC := IC + 2
1662:    END (*GENWORD*) ;
1663:
1664:    PROCEDURE GENBIG(IVAL: INTEGER);
1665:      VAR LOWORDER: CHAR;
1666:    BEGIN
1667:      IF IVAL <= 127 THEN GENBYTE(IVAL)
1668:      ELSE
1669:        BEGIN MOVELEFT(IVAL,CODEP^[IC],2); LOWORDER := CODEP^[IC];
1670:          CODEP^[IC] := CHR(ORD(CODEP^[IC+1])+128);
1671:          CODEP^[IC+1] := LOWORDER; IC := IC+2
1672:        END
1673:    END (*GENBIG*) ;
1674:
1675:    PROCEDURE GEN0(FOP: OPRANGE);
1676:      VAR I: INTEGER;
1677:    BEGIN
1678:      GENBYTE(FOP+128);
1679:      IF FOP = 38(*LCA*) THEN
1680:        WITH GATTR.CVAL.VALP^ DO
1681:          BEGIN GENBYTE(SLGTH);
1682:            FOR I := 1 TO SLGTH DO GENBYTE(ORD(SVAL[I]))
1683:          END
1684:    END (*GEN0*) ;
1685:
1686:    PROCEDURE GEN1(FOP: OPRANGE; FP2: INTEGER);
1687:      LABEL 1;
1688:      VAR I,J: INTEGER;
1689:    BEGIN
1690:      GENBYTE(FOP+128);
1691:      IF FOP = 51(*LDC*) THEN
1692:        BEGIN
1693:          IF FP2 = 2 THEN I := REALSIZE
1694:          ELSE
1695:            BEGIN I := 8;
1696:              WHILE I > 0 DO
1697:                IF GATTR.CVAL.VALP^.CSTVAL[I] <> 0 THEN GOTO 1
1698:                ELSE I := I - 1;
1699:        1:  END;
1700:          GATTR.TYPTR^.SIZE := I;
1701:          IF I > 1 THEN
1702:            BEGIN GENBYTE(I);
1703:              FOR J := I DOWNTO 1 DO GENWORD(GATTR.CVAL.VALP^.CSTVAL[J])
1704:            END
1705:          ELSE
1706:            BEGIN IC := IC - 1;
1707:              IF I = 1 THEN GENLDC(GATTR.CVAL.VALP^.CSTVAL[1])
1708:            END
1709:        END
1710:      ELSE
1711:        IF FOP IN [30(*CSP*),32(*ADJ*),45(*RNP*),
1712:                   46(*CIP*),60(*LDM*),61(*STM*),
1713:                   65(*RBP*),66(*CBP*),78(*CLP*),
1714:                   42(*SAS*),79(*CGP*)] THEN GENBYTE(FP2)
1715:        ELSE
1716:          IF ((FOP = 74(*LDL*)) OR (FOP = 39(*LDO*)))
```

```
1717:             AND (FP2 <= 16) THEN
1718:           BEGIN IC := IC-1;
1719:             IF FOP = 39(*LDO*) THEN GENBYTE(231+FP2)
1720:             ELSE GENBYTE(215+FP2)
1721:           END
1722:         ELSE
1723:           IF (FOP = 35(*IND*)) AND (FP2 <= 7) THEN
1724:             BEGIN IC := IC-1; GENBYTE(248+FP2) END
1725:           ELSE GENBIG(FP2)
1726:   END (*GEN1*) ;
1727:
1728:   PROCEDURE GEN2(FOP: OPRANGE; FP1,FP2: INTEGER);
1729:   BEGIN
1730:     IF (FOP = 64(*IXP*)) OR (FOP = 77(*CXP*)) THEN
1731:       BEGIN GENBYTE(FOP+128); GENBYTE(FP1); GENBYTE(FP2);
1732:       END
1733:     ELSE
1734:       IF FOP IN [47(*EQU*),48(*GEQ*),49(*GRT*),
1735:                  52(*LEQ*),53(*LES*),55(*NEQ*)] THEN
1736:         IF FP1 = 0 THEN GEN0(FOP+20)
1737:         ELSE
1738:           BEGIN GEN1(FOP,FP1+FP1);
1739:             IF FP1 > 4 THEN GENBIG(FP2)
1740:           END
1741:       ELSE
1742:         BEGIN (*LDA,LOD,STR*)
1743:           IF FP1 = 0 THEN GEN1(FOP+20,FP2)
1744:           ELSE
1745:             BEGIN
1746:               GENBYTE(FOP+128); GENBYTE(FP1); GENBIG(FP2)
1747:             END
1748:         END;
1749:   END (*GEN2*) ;
1750:
1751:   PROCEDURE GENLDC;
1752:   BEGIN
1753:     IF (IVAL >= 0) AND (IVAL <= 127) THEN GENBYTE(IVAL)
1754:     ELSE
1755:       BEGIN GENBYTE(51(*LDC*)+148);
1756:         MOVELEFT(IVAL,CODEP^[IC],2);
1757:         IC := IC+2
1758:       END
1759:   END (*GENLDC*) ;
1760:
1761:   PROCEDURE GENJMP(FOP: OPRANGE; FLBP: LBP);
1762:     VAR DISP: INTEGER;
1763:   BEGIN
1764:     WITH FLBP^ DO
1765:       IF DEFINED THEN
1766:         BEGIN
1767:           GENBYTE(FOP+128);
1768:           DISP := OCCURIC-IC-1;
1769:           IF (DISP >= 0) AND (DISP <= 127) THEN GENBYTE(DISP)
1770:           ELSE
1771:             BEGIN
1772:               IF JTABINX = 0 THEN
1773:                 BEGIN JTABINX := NEXTJTAB;
1774:                   IF NEXTJTAB = MAXJTAB THEN ERROR(253)
1775:                   ELSE NEXTJTAB := NEXTJTAB + 1;
1776:                   JTAB[JTABINX] := OCCURIC
1777:                 END;
1778:               DISP := -JTABINX;
1779:               GENBYTE(248-JTABINX-JTABINX)
1780:             END;
1781:         END
1782:       ELSE
```

```
1783:          BEGIN MOVELEFT(REFLIST,CODEP^[IC],2);
1784:            IF FOP = 57(*UJP*) THEN DISP := IC + 4096
1785:            ELSE DISP := IC;
1786:            REFLIST := DISP; IC := IC+2
1787:          END;
1788:    END (*GENJMP*) ;
1789:
1790:    PROCEDURE LOAD; FORWARD;
1791:
1792:    PROCEDURE GENFJP(FLBP: LBP);
1793:    BEGIN LOAD;
1794:      IF GATTR.TYPTR <> BOOLPTR THEN ERROR(135);
1795:      GENJMP(33(*FJP*),FLBP)
1796:    END (*GENFJP*) ;
1797:
1798:    PROCEDURE GENLABEL(VAR FLBP: LBP);
1799:    BEGIN NEW(FLBP);
1800:      WITH FLBP^ DO
1801:        BEGIN DEFINED := FALSE; REFLIST := MAXADDR END
1802:    END (*GENLABEL*) ;
1803:
1804:    PROCEDURE PUTLABEL(FLBP: LBP);
1805:      VAR LREF: INTEGER; LOP: OPRANGE;
1806:    BEGIN
1807:      WITH FLBP^ DO
1808:        BEGIN LREF := REFLIST;
1809:          DEFINED := TRUE; OCCURIC := IC; JTABINX := 0;
1810:          WHILE LREF < MAXADDR DO
1811:            BEGIN
1812:              IF LREF >= 4096 THEN
1813:                BEGIN LREF := LREF - 4096; LOP := 57(*UJP*) END
1814:              ELSE LOP := 33(*FJP*);
1815:              IC := LREF;
1816:              MOVELEFT(CODEP^[IC],LREF,2);
1817:              GENJMP(LOP,FLBP)
1818:            END;
1819:          IC := OCCURIC
1820:        END
1821:    END (*PUTLABEL*) ;
1822:
1823:    PROCEDURE LOAD;
1824:    BEGIN
1825:      WITH GATTR DO
1826:        IF TYPTR <> NIL THEN
1827:          BEGIN
1828:            CASE KIND OF
1829:              CST:   IF (TYPTR^.FORM = SCALAR) AND (TYPTR <> REALPTR) THEN
1830:                       GENLDC(CVAL.IVAL)
1831:                     ELSE
1832:                       IF TYPTR = NILPTR THEN GEN0(31(*LDCN*))
1833:                       ELSE
1834:                         IF TYPTR = REALPTR THEN GEN1(51(*LDC*),2)
1835:                         ELSE GEN1(51(*LDC*),5);
1836:              VARBL: CASE ACCESS OF
1837:                       DRCT:   IF VLEVEL = 1 THEN GEN1(39(*LDO*),DPLMT)
1838:                               ELSE GEN2(54(*LOD*),LEVEL-VLEVEL,DPLMT);
1839:                       INDRCT: GEN1(35(*IND*),IDPLMT);
1840:                       PACKD:  GEN0(58(*LDP*));
1841:                       MULTI:  GEN1(60(*LDM*),TYPTR^.SIZE);
1842:                       BYTE:   GEN0(62(*LDB*))
1843:                     END;
1844:              EXPR:
1845:            END;
1846:            IF (TYPTR^.FORM = POWER) AND (KIND <> EXPR) THEN
1847:              GENLDC(TYPTR^.SIZE);
1848:            KIND := EXPR
```

```
1849:          END
1850:    END (*LOAD*) ;
1851:
1852:    PROCEDURE STORE(VAR FATTR: ATTR);
1853:    BEGIN
1854:      WITH FATTR DO
1855:        IF TYPTR <> NIL THEN
1856:          CASE ACCESS OF
1857:            DRCT:   IF VLEVEL = 1 THEN GEN1(43(*SRO*),DPLMT)
1858:                    ELSE GEN2(56(*STR*),LEVEL-VLEVEL,DPLMT);
1859:            INDRCT: IF IDPLMT <> 0 THEN ERROR(400)
1860:                    ELSE GEN0(26(*STO*));
1861:            PACKD:  GEN0(59(*STP*));
1862:            MULTI:  GEN1(61(*STM*),TYPTR^.SIZE);
1863:            BYTE:   GEN0(63(*STB*))
1864:          END
1865:    END (*STORE*) ;
1866:
1867:    PROCEDURE LOADADDRESS;
1868:    BEGIN
1869:      WITH GATTR DO
1870:        IF TYPTR <> NIL THEN
1871:          BEGIN
1872:            CASE KIND OF
1873:              CST:   IF STRGTYPE(TYPTR) THEN GEN0(38(*LCA*))
1874:                     ELSE ERROR(400);
1875:              VARBL: CASE ACCESS OF
1876:                       DRCT:   IF VLEVEL = 1 THEN GEN1(37(*LAO*),DPLMT)
1877:                               ELSE GEN2(50(*LDA*),LEVEL-VLEVEL,DPLMT);
1878:                       INDRCT: IF IDPLMT <> 0 THEN GEN1(34(*INC*),IDPLMT+IDPLMT);
1879:                       PACKD:  ERROR(103)
1880:                     END
1881:            END;
1882:            KIND := VARBL; ACCESS := INDRCT; IDPLMT := 0
1883:          END
1884:    END (*LOADADDRESS*) ;
1885:
1886:    PROCEDURE WRITECODE(FORCEBUF: BOOLEAN);
1887:      VAR CODEINX,LIC,I: INTEGER;
1888:    BEGIN CODEINX := 0; LIC := IC;
1889:      REPEAT
1890:        I := 512-CURBYTE;
1891:        IF I > LIC THEN I := LIC;
1892:        MOVELEFT(CODEP^[CODEINX],DISKBUF[CURBYTE],I);
1893:        CODEINX := CODEINX+I;
1894:        CURBYTE := CURBYTE+I;
1895:        IF (CURBYTE = 512) OR FORCEBUF THEN
1896:          BEGIN
1897:            IF BLOCKWRITE(USERINFO.WORKCODE^,DISKBUF,1,CURBLK) <> 1 THEN
1898:              ERROR(402);
1899:            CURBLK := CURBLK+1; CURBYTE := 0
1900:          END;
1901:        LIC := LIC-I
1902:      UNTIL LIC = 0;
1903:    END (*WRITECODE*) ;
1904:
1905:    PROCEDURE FINISHSEG;
1906:      VAR I: INTEGER;
1907:    BEGIN IC := 0;
1908:      FOR I := NEXTPROC-1 DOWNTO 1 DO GENWORD(SEGINX+IC-PROCTABLE[I]);
1909:      GENBYTE(SEG); GENBYTE(NEXTPROC-1);
1910:      SEGTABLE[SEG].CODELENG := SEGINX+IC;
1911:      WRITECODE(TRUE); SEGINX := 0; CODEINSEG := FALSE
1912:    END (*FINISHSEG*) ;
1913:
1914:  (*$I XCOMP:D.TEXT *)
```

```
1915:
1916:    PROCEDURE EXPRESSION(FSYS: SETOFSYS); FORWARD;
1917:
1918:    PROCEDURE SELECTOR(FSYS: SETOFSYS; FCP: CTP);
1919:      VAR LATTR: ATTR; LCP: CTP; LMIN,LMAX: INTEGER;
1920:    BEGIN
1921:      WITH FCP^, GATTR DO
1922:        BEGIN TYPTR := IDTYPE; KIND := VARBL;
1923:          CASE KLASS OF
1924:            VARS:
1925:              IF VKIND = ACTUAL THEN
1926:                BEGIN ACCESS := DRCT; VLEVEL := VLEV;
1927:                  DPLMT := VADDR
1928:                END
1929:              ELSE
1930:                BEGIN
1931:                  IF VLEV = 1 THEN GEN1(39(*LDO*),VADDR)
1932:                  ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR);
1933:                  ACCESS := INDRCT; IDPLMT := 0
1934:                END;
1935:            FIELD:
1936:              WITH DISPLAY[DISX] DO
1937:               BEGIN
1938:                IF OCCUR = CREC THEN
1939:                  BEGIN ACCESS := DRCT; VLEVEL := CLEV;
1940:                    DPLMT := CDSPL + FLDADDR
1941:                  END
1942:                ELSE
1943:                  BEGIN
1944:                    IF LEVEL = 1 THEN GEN1(39(*LDO*),VDSPL)
1945:                    ELSE GEN2(54(*LOD*),0,VDSPL);
1946:                    ACCESS := INDRCT; IDPLMT := FLDADDR
1947:                  END;
1948:                IF FISPACKD THEN
1949:                  BEGIN LOADADDRESS;
1950:                    IF ((FLDRBIT = 0) OR (FLDRBIT = 8))
1951:                          AND (FLDWIDTH = 8) THEN
1952:                      BEGIN ACCESS := BYTE;
1953:                        IF FLDRBIT = 8 THEN GEN1(34(*INC*),1)
1954:                      END
1955:                    ELSE
1956:                      BEGIN ACCESS := PACKD;
1957:                        GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
1958:                      END
1959:                  END
1960:               END;
1961:            FUNC:
1962:              IF PFDECKIND <> DECLARED THEN ERROR(150)
1963:              ELSE
1964:                IF NOT INSCOPE THEN ERROR(103)
1965:                ELSE
1966:                    BEGIN ACCESS := DRCT; VLEVEL := PFLEV + 1;
1967:                      DPLMT := LCAFTERMARKSTACK
1968:                    END
1969:          END (*CASE*);
1970:          IF TYPTR <> NIL THEN
1971:            IF (TYPTR^.FORM <= POWER) AND
1972:               (TYPTR^.SIZE > PTRSIZE) THEN
1973:              BEGIN LOADADDRESS; ACCESS := MULTI END
1974:        END (*WITH*);
1975:      IF NOT (SY IN SELECTSYS + FSYS) THEN
1976:        BEGIN ERROR(59); SKIP(SELECTSYS + FSYS) END;
1977:      WHILE SY IN SELECTSYS DO
1978:        BEGIN
1979:    (*[*) IF SY = LBRACK THEN
1980:            BEGIN
```

```
1981:              REPEAT LATTR := GATTR;
1982:                WITH LATTR DO
1983:                  IF TYPTR <> NIL THEN
1984:                    IF TYPTR^.FORM <> ARRAYS THEN
1985:                      BEGIN ERROR(138); TYPTR := NIL END;
1986:                LOADADDRESS;
1987:                INSYMBOL; EXPRESSION(FSYS + [COMMA,RBRACK]);
1988:                LOAD;
1989:                IF GATTR.TYPTR <> NIL THEN
1990:                  IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(113);
1991:                IF LATTR.TYPTR <> NIL THEN
1992:                  WITH LATTR.TYPTR^ DO
1993:                    BEGIN
1994:                      IF COMPTYPES(INXTYPE,GATTR.TYPTR) THEN
1995:                        BEGIN
1996:                          IF (INXTYPE <> NIL) AND
1997:                              NOT STRGTYPE(LATTR.TYPTR) THEN
1998:                            BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
1999:                              IF RANGECHECK THEN
2000:                                BEGIN GENLDC(LMIN); GENLDC(LMAX);
2001:                                  GEN0(8(*CHK*))
2002:                                END;
2003:                              IF LMIN <> 0 THEN
2004:                                BEGIN GENLDC(ABS(LMIN));
2005:                                  IF LMIN > 0 THEN GEN0(21(*SBI*))
2006:                                  ELSE GEN0(2(*ADI*))
2007:                                END
2008:                            END
2009:                        END
2010:                      ELSE ERROR(139);
2011:                      WITH GATTR DO
2012:                        BEGIN TYPTR := AELTYPE; KIND := VARBL;
2013:                          ACCESS := INDRCT; IDPLMT := 0;
2014:                          IF TYPTR <> NIL THEN
2015:                            IF AISPACKD THEN
2016:                              IF ELWIDTH = 8 THEN
2017:                                BEGIN ACCESS := BYTE;
2018:                                  IF STRGTYPE(LATTR.TYPTR) AND RANGECHECK THEN
2019:                                    GEN0(27(*IXS*))
2020:                                  ELSE GEN0(2(*ADI*))
2021:                                END
2022:                              ELSE
2023:                                BEGIN ACCESS := PACKD;
2024:                                  GEN2(64(*IXP*),ELSPERWD,ELWIDTH)
2025:                                END
2026:                            ELSE
2027:                              BEGIN GEN1(36(*IXA*),TYPTR^.SIZE);
2028:                                IF (TYPTR^.FORM <= POWER) AND
2029:                                   (TYPTR^.SIZE > PTRSIZE) THEN
2030:                                  ACCESS := MULTI
2031:                                END
2032:                          END
2033:                      END
2034:                UNTIL SY <> COMMA;
2035:                IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
2036:              END (*IF SY = LBRACK*)
2037:            ELSE
2038:        (*.*)   IF SY = PERIOD THEN
2039:                  BEGIN
2040:                    WITH GATTR DO
2041:                      BEGIN
2042:                        IF TYPTR <> NIL THEN
2043:                          IF TYPTR^.FORM <> RECORDS THEN
2044:                            BEGIN ERROR(140); TYPTR := NIL END;
2045:                        INSYMBOL;
2046:                        IF SY = IDENT THEN
```

```
2047:                     BEGIN
2048:                       IF TYPTR <> NIL THEN
2049:                         BEGIN SEARCHSECTION(TYPTR^.FSTFLD,LCP);
2050:                           IF LCP = NIL THEN
2051:                             BEGIN ERROR(152); TYPTR := NIL END
2052:                           ELSE
2053:                             WITH LCP^ DO
2054:                               BEGIN TYPTR := IDTYPE;
2055:                                 CASE ACCESS OF
2056:                                   DRCT:   DPLMT := DPLMT + FLDADDR;
2057:                                   INDRCT: IDPLMT := IDPLMT + FLDADDR;
2058:                                   MULTI,BYTE,
2059:                                   PACKD:  ERROR(400)
2060:                                 END (*CASE ACCESS*);
2061:                                 IF FISPACKD THEN
2062:                                   BEGIN LOADADDRESS;
2063:                                     IF ((FLDRBIT = 0) OR (FLDRBIT = 8))
2064:                                         AND (FLDWIDTH = 8) THEN
2065:                                       BEGIN ACCESS := BYTE;
2066:                                         IF FLDRBIT = 8 THEN GEN1(34(*INC*),1)
2067:                                       END
2068:                                     ELSE
2069:                                       BEGIN ACCESS := PACKD;
2070:                                         GENLDC(FLDWIDTH); GENLDC(FLDRBIT)
2071:                                       END
2072:                                   END;
2073:                                 IF TYPTR <> NIL THEN
2074:                                   IF (TYPTR^.FORM <= POWER) AND
2075:                                      (TYPTR^.SIZE > PTRSIZE) THEN
2076:                                     BEGIN LOADADDRESS; ACCESS := MULTI END
2077:                               END
2078:                         END;
2079:                       INSYMBOL
2080:                     END (*SY = IDENT*)
2081:                   ELSE ERROR(2)
2082:                 END (*WITH GATTR*)
2083:             END (*IF SY = PERIOD*)
2084:           ELSE
2085:     (*^*)     BEGIN
2086:                 IF GATTR.TYPTR <> NIL THEN
2087:                   WITH GATTR,TYPTR^ DO
2088:                     IF (FORM = POINTER) OR (FORM = FILES) THEN
2089:                       BEGIN LOAD; KIND := VARBL;
2090:                         ACCESS := INDRCT; IDPLMT := 0;
2091:                         IF FORM = POINTER THEN TYPTR := ELTYPE
2092:                         ELSE
2093:                           BEGIN TYPTR := FILTYPE;
2094:                             IF TYPTR = NIL THEN ERROR(399)
2095:                           END;
2096:                         IF TYPTR <> NIL THEN
2097:                           IF (TYPTR^.FORM <= POWER) AND
2098:                              (TYPTR^.SIZE > PTRSIZE) THEN
2099:                                ACCESS := MULTI
2100:                       END
2101:                     ELSE ERROR(141);
2102:                 INSYMBOL
2103:               END;
2104:           IF NOT (SY IN FSYS + SELECTSYS) THEN
2105:             BEGIN ERROR(6); SKIP(FSYS + SELECTSYS) END
2106:         END (*WHILE*)
2107:     END (*SELECTOR*) ;
2108:
2109:     PROCEDURE CALL(FSYS: SETOFSYS; FCP: CTP);
2110:       VAR LKEY: 1..40; WASLPARENT: BOOLEAN;
2111:
2112:       PROCEDURE VARIABLE(FSYS: SETOFSYS);
```

```
2113:       VAR LCP: CTP;
2114:     BEGIN
2115:       IF SY = IDENT THEN
2116:         BEGIN SEARCHID([FIELD,VARS],LCP); INSYMBOL END
2117:       ELSE BEGIN ERROR(2); LCP := UVARPTR END;
2118:       SELECTOR(FSYS,LCP)
2119:     END (*VARIABLE*) ;
2120:
2121:     PROCEDURE STRGVAR(FSYS: SETOFSYS; MUSTBEVAR: BOOLEAN);
2122:     BEGIN EXPRESSION(FSYS);
2123:       WITH GATTR DO
2124:         IF ((KIND = CST) AND (TYPTR = CHARPTR))
2125:             OR STRGTYPE(TYPTR) THEN
2126:           IF KIND = VARBL THEN LOADADDRESS
2127:           ELSE
2128:             BEGIN
2129:               IF MUSTBEVAR THEN ERROR(154);
2130:               IF KIND = CST THEN
2131:                 BEGIN
2132:                   IF TYPTR = CHARPTR THEN
2133:                     BEGIN
2134:                       WITH SCONST^ DO
2135:                         BEGIN CCLASS := STRG; SLGTH := 1;
2136:                           SVAL[1] := CHR(CVAL.IVAL)
2137:                         END;
2138:                       CVAL.VALP := SCONST;
2139:                       NEW(TYPTR,ARRAYS,TRUE,TRUE);
2140:                       TYPTR^ := STRGPTR^;
2141:                       TYPTR^.MAXLENG := 1
2142:                     END;
2143:                   LOADADDRESS
2144:                 END
2145:             END
2146:         ELSE
2147:           BEGIN
2148:             IF GATTR.TYPTR <> NIL THEN ERROR(125);
2149:             GATTR.TYPTR := STRGPTR
2150:           END
2151:     END (*STRGVAR*) ;
2152:
2153:     PROCEDURE NEWSTMT;
2154:       LABEL 1;
2155:       VAR LSP,LSP1: STP; VARTS,LMIN,LMAX: INTEGER;
2156:           LSIZE,LSZ: ADDRRANGE; LVAL: VALU;
2157:     BEGIN VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2158:       LSP := NIL; VARTS := 0; LSIZE := 0;
2159:       IF GATTR.TYPTR <> NIL THEN
2160:         WITH GATTR.TYPTR^ DO
2161:           IF FORM = POINTER THEN
2162:             BEGIN
2163:               IF ELTYPE <> NIL THEN
2164:                 WITH ELTYPE^ DO
2165:                   BEGIN LSIZE := SIZE;
2166:                     IF FORM = RECORDS THEN LSP := RECVAR
2167:                   END
2168:             END
2169:           ELSE ERROR(116);
2170:       WHILE SY = COMMA DO
2171:         BEGIN INSYMBOL;
2172:           CONSTANT(FSYS + [COMMA,RPARENT],LSP1,LVAL);
2173:           VARTS := VARTS + 1;
2174:           IF LSP = NIL THEN ERROR(158)
2175:           ELSE
2176:             IF LSP^.FORM <> TAGFLD THEN ERROR(162)
2177:             ELSE
2178:               IF LSP^.TAGFIELDP <> NIL THEN
```

```
2179:                 IF STRGTYPE(LSP1) OR (LSP1 = REALPTR) THEN ERROR(159)
2180:                 ELSE
2181:                   IF COMPTYPES(LSP^.TAGFIELDP^.IDTYPE,LSP1) THEN
2182:                     BEGIN
2183:                       LSP1 := LSP^.FSTVAR;
2184:                       WHILE LSP1 <> NIL DO
2185:                         WITH LSP1^ DO
2186:                           IF VARVAL.IVAL = LVAL.IVAL THEN
2187:                             BEGIN LSIZE := SIZE; LSP := SUBVAR;
2188:                               GOTO 1
2189:                             END
2190:                           ELSE LSP1 := NXTVAR;
2191:                       LSIZE := LSP^.SIZE; LSP := NIL;
2192:                     END
2193:                   ELSE ERROR(116);
2194:     1:  END (*WHILE*) ;
2195:       GENLDC(LSIZE);
2196:       GEN1(30(*CSP*),1(*NEW*))
2197:     END (*NEWSTMT*) ;
2198:
2199:     PROCEDURE MOVE;
2200:     BEGIN VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2201:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2202:       IF LKEY = 27 THEN
2203:         BEGIN EXPRESSION(FSYS + [COMMA]); LOAD END
2204:       ELSE
2205:         BEGIN VARIABLE(FSYS + [COMMA]); LOADADDRESS END;
2206:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2207:       EXPRESSION(FSYS + [RPARENT]); LOAD;
2208:       IF LKEY = 27 THEN GEN1(30(*CSP*),10(*FLC*))
2209:       ELSE
2210:         IF LKEY = 21 THEN GEN1(30(*CSP*),2(*MVL*))
2211:         ELSE GEN1(30(*CSP*),3(*MVR*))
2212:     END (*MOVE*) ;
2213:
2214:     PROCEDURE EXIT;
2215:       VAR LCP: CTP;
2216:     BEGIN
2217:       IF SY = IDENT THEN
2218:         BEGIN SEARCHID([PROC,FUNC],LCP); INSYMBOL END
2219:       ELSE
2220:         IF (SY = PROGSY) THEN
2221:           BEGIN LCP := OUTERBLOCK; INSYMBOL END
2222:         ELSE LCP := NIL;
2223:       IF LCP <> NIL THEN
2224:         IF LCP^.PFDECKIND = DECLARED THEN
2225:           BEGIN GENLDC(LCP^.PFSEG); GENLDC(LCP^.PFNAME) END
2226:         ELSE ERROR(125)
2227:       ELSE ERROR(125);
2228:       GEN1(30(*CSP*),4(*XIT*))
2229:     END (*EXIT*) ;
2230:
2231:     PROCEDURE UNITIO;
2232:     BEGIN
2233:       IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2234:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2235:       VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2236:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2237:       EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2238:       IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2239:       IF SY = COMMA THEN
2240:         BEGIN INSYMBOL;
2241:           IF SY = COMMA THEN GENLDC(0)
2242:           ELSE
2243:             BEGIN
2244:               EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
```

```
2245:               IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2246:             END
2247:         END
2248:       ELSE GENLDC(0);
2249:       IF SY = COMMA THEN
2250:         BEGIN INSYMBOL;
2251:           EXPRESSION(FSYS + [RPARENT]); LOAD;
2252:           IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2253:         END
2254:       ELSE GENLDC(0);
2255:       IF LKEY = 13 THEN GEN1(30(*CSP*),5(*URD*))
2256:       ELSE GEN1(30(*CSP*),6(*UWT*))
2257:     END (*UNITIO*);
2258:
2259:     PROCEDURE CONCAT;
2260:       VAR LLC: ADDRRANGE; TEMPLGTH: INTEGER;
2261:     BEGIN TEMPLGTH := 0;
2262:       LLC := LC; LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2263:       GENLDC(0); GEN2(56(*STR*),0,LLC);
2264:       GEN2(50(*LDA*),0,LLC);
2265:       REPEAT
2266:         STRGVAR(FSYS + [COMMA,RPARENT],FALSE);
2267:         TEMPLGTH := TEMPLGTH + GATTR.TYPTR^.MAXLENG;
2268:         IF TEMPLGTH < STRGLGTH THEN GENLDC(TEMPLGTH)
2269:         ELSE GENLDC(STRGLGTH);
2270:         GEN2(77(*CXP*),0(*SYS*),23(*SCONCAT*));
2271:         GEN2(50(*LDA*),0,LLC);
2272:         TEST := SY <> COMMA;
2273:         IF NOT TEST THEN INSYMBOL
2274:       UNTIL TEST;
2275:       IF TEMPLGTH < STRGLGTH THEN
2276:         LC := LLC + (TEMPLGTH DIV CHRSPERWD) + 1
2277:       ELSE TEMPLGTH := STRGLGTH;
2278:       IF LC > LCMAX THEN LCMAX := LC;
2279:       LC := LLC;
2280:       WITH GATTR DO
2281:         BEGIN NEW(TYPTR,ARRAYS,TRUE,TRUE);
2282:           TYPTR^ := STRGPTR^;
2283:           TYPTR^.MAXLENG := TEMPLGTH
2284:         END
2285:     END (*CONCAT*) ;
2286:
2287:     PROCEDURE COPYDELETE;
2288:       VAR LLC: ADDRRANGE; LSP: STP;
2289:     BEGIN
2290:       IF LKEY = 19 THEN
2291:         BEGIN LLC := LC;
2292:           LC := LC + (STRGLGTH DIV CHRSPERWD) + 1;
2293:         END;
2294:       STRGVAR(FSYS + [COMMA], LKEY = 18);
2295:       IF LKEY = 19 THEN
2296:         BEGIN LSP := GATTR.TYPTR;
2297:           GEN2(50(*LDA*),0,LLC)
2298:         END;
2299:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2300:       EXPRESSION(FSYS + [COMMA]); LOAD;
2301:       IF GATTR.TYPTR <> NIL THEN
2302:         IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2303:       IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2304:       EXPRESSION(FSYS + [RPARENT]); LOAD;
2305:       IF GATTR.TYPTR <> NIL THEN
2306:         IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2307:       IF LKEY = 19 THEN
2308:         BEGIN
2309:           GEN2(77(*CXP*),0(*SYS*),25(*SCOPY*));
2310:           GEN2(50(*LDA*),0,LLC);
```

```
2311:            IF LSP^.MAXLENG < STRGLGTH THEN
2312:              LC := LLC + (LSP^.MAXLENG DIV CHRSPERWD) + 1;
2313:            IF LC > LCMAX THEN LCMAX := LC;
2314:            LC := LLC; GATTR.TYPTR := LSP
2315:          END
2316:        ELSE GEN2(77(*CXP*),0(*SYS*),26(*SDELETE*))
2317:      END (*COPYDELETE*) ;
2318:
2319:      PROCEDURE CLOSE;
2320:      BEGIN
2321:        VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2322:        IF GATTR.TYPTR <> NIL THEN
2323:          IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
2324:        IF SY = COMMA THEN
2325:          BEGIN INSYMBOL;
2326:            IF SY = IDENT THEN
2327:             BEGIN
2328:              IF ID = 'NORMAL  ' THEN GENLDC(0)
2329:              ELSE
2330:                IF ID = 'LOCK    ' THEN GENLDC(1)
2331:                ELSE
2332:                  IF ID = 'PURGE   ' THEN GENLDC(2)
2333:                  ELSE
2334:                    IF ID = 'CRUNCH  ' THEN GENLDC(3)
2335:                    ELSE ERROR(2);
2336:              INSYMBOL
2337:             END
2338:            ELSE ERROR(2)
2339:          END
2340:        ELSE GENLDC(0);
2341:        GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
2342:        IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2343:      END (*CLOSE*) ;
2344:
2345:      PROCEDURE GETPUTETC;
2346:      BEGIN
2347:        VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2348:        IF GATTR.TYPTR <> NIL THEN
2349:          IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2350:        CASE LKEY OF
2351:          32:  BEGIN
2352:                 IF SY = COMMA THEN
2353:                   BEGIN
2354:                     INSYMBOL; EXPRESSION(FSYS + [RPARENT]); LOAD;
2355:                     IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2356:                   END
2357:                 ELSE ERROR(125);
2358:                 GEN2(77(*CXP*),0(*SYS*),9(*FSEEK*))
2359:               END;
2360:          33:  GEN2(77(*CXP*),0(*SYS*),4(*FRESET*));
2361:          34:  GEN2(77(*CXP*),0(*SYS*),7(*FGET*));
2362:          35:  GEN2(77(*CXP*),0(*SYS*),8(*FPUT*));
2363:          40:  BEGIN
2364:                 IF GATTR.TYPTR <> NIL THEN
2365:                   IF GATTR.TYPTR^.FILTYPE <> CHARPTR THEN ERROR(399);
2366:                 GENLDC(12); GENLDC(0);
2367:                 GEN2(77(*CXP*),0(*SYS*),17(*WRC*))
2368:               END
2369:        END (*CASE*) ;
2370:        IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2371:      END (*GETPUTETC*) ;
2372:
2373:      PROCEDURE SCAN;
2374:      BEGIN
2375:        IF GATTR.TYPTR <> NIL THEN
2376:          IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
```

```
2377:        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2378:        IF SY = RELOP THEN
2379:          BEGIN
2380:            IF OP = EQOP THEN GENLDC(0)
2381:            ELSE
2382:              IF OP = NEOP THEN GENLDC(1)
2383:              ELSE ERROR(125);
2384:            INSYMBOL
2385:          END
2386:        ELSE ERROR(125);
2387:        EXPRESSION(FSYS + [COMMA]); LOAD;
2388:        IF GATTR.TYPTR <> NIL THEN
2389:          IF GATTR.TYPTR <> CHARPTR THEN ERROR(125);
2390:        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2391:        VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2392:        IF SY = COMMA THEN
2393:          BEGIN INSYMBOL;
2394:            EXPRESSION(FSYS + [RPARENT]); LOAD
2395:          END
2396:        ELSE GENLDC(0);
2397:        GEN1(30(*CSP*),11(*SCN*));
2398:        GATTR.TYPTR := INTPTR
2399:      END (*SCAN*) ;
2400:
2401:      PROCEDURE BLOCKIO;
2402:      BEGIN
2403:        VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2404:        IF GATTR.TYPTR <> NIL THEN
2405:          IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
2406:          ELSE
2407:            IF GATTR.TYPTR^.FILTYPE <> NIL THEN ERROR(399);
2408:        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2409:        VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2410:        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2411:        EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2412:        IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2413:        IF SY = COMMA THEN
2414:          BEGIN INSYMBOL;
2415:            EXPRESSION(FSYS + [RPARENT]); LOAD;
2416:            IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2417:          END
2418:        ELSE GENLDC(-1);
2419:        IF LKEY = 37 THEN GENLDC(1) ELSE GENLDC(0);
2420:        GENLDC(0); GENLDC(0);
2421:        GEN2(77(*CXP*),0(*SYS*),28(*BLOCKIO*));
2422:        IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
2423:        GATTR.TYPTR := INTPTR
2424:      END (*BLOCKIO*) ;
2425:
2426:      PROCEDURE DRAWSTUFF;
2427:        VAR I,N: INTEGER;
2428:      BEGIN
2429:        VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2430:        IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2431:        VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2432:        IF LKEY = 42 THEN N := 6
2433:        ELSE N := 5;
2434:        FOR I := 0 TO N DO
2435:          BEGIN
2436:            IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2437:            EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2438:            IF GATTR.TYPTR <> NIL THEN
2439:              IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2440:          END;
2441:        IF LKEY = 42 THEN N := 13
2442:        ELSE N := 12;
```

```
2443:         GEN1(30(*CSP*),N)
2444:       END (*DRAWSTUFF*) ;
2445:
2446:       PROCEDURE SIZEOF;
2447:         VAR LCP: CTP;
2448:       BEGIN
2449:         IF SY = IDENT THEN
2450:           BEGIN SEARCHID([TYPES,VARS,FIELD],LCP); INSYMBOL;
2451:             IF LCP^.IDTYPE <> NIL THEN
2452:               GENLDC(LCP^.IDTYPE^.SIZE*CHRSPERWD)
2453:           END;
2454:         GATTR.TYPTR := INTPTR
2455:       END (*SIZEOF*) ;
2456:
2457:
2458:       PROCEDURE LOADIDADDR(FCP: CTP);
2459:       BEGIN
2460:           WITH FCP^ DO
2461:             IF VKIND = ACTUAL THEN
2462:               IF VLEV = 1 THEN GEN1(37(*LAO*),VADDR)
2463:               ELSE GEN2(50(*LDA*),LEVEL-VLEV,VADDR)
2464:             ELSE
2465:               IF VLEV = 1 THEN GEN1(39(*LDO*),VADDR)
2466:               ELSE GEN2(54(*LOD*),LEVEL-VLEV,VADDR)
2467:       END (*LOADIDADDR*) ;
2468:
2469:       PROCEDURE READ;
2470:         VAR FILEPTR,LCP: CTP;
2471:       BEGIN FILEPTR := INPUTPTR;
2472:         IF (SY = IDENT) AND WASLPARENT THEN
2473:           BEGIN SEARCHID([FIELD,VARS],LCP);
2474:             IF LCP^.IDTYPE <> NIL THEN
2475:               IF LCP^.IDTYPE^.FORM = FILES THEN
2476:                 IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
2477:                   BEGIN INSYMBOL; FILEPTR := LCP;
2478:                     IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
2479:                     IF SY = COMMA THEN INSYMBOL
2480:                   END
2481:           END
2482:         ELSE
2483:           IF WASLPARENT THEN ERROR(2);
2484:         IF WASLPARENT AND (SY <> RPARENT) THEN
2485:           BEGIN
2486:             REPEAT LOADIDADDR(FILEPTR);
2487:               VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2488:               IF GATTR.TYPTR <> NIL THEN
2489:                 IF COMPTYPES(INTPTR,GATTR.TYPTR) THEN
2490:                   GEN2(77(*CXP*),0(*SYS*),12(*FRDI*))
2491:                 ELSE
2492:                   IF COMPTYPES(REALPTR,GATTR.TYPTR) THEN
2493:                     GEN2(77(*CXP*),0(*SYS*),14(*FRDR*))
2494:                   ELSE
2495:                     IF COMPTYPES(CHARPTR,GATTR.TYPTR) THEN
2496:                       GEN2(77(*CXP*),0(*SYS*),16(*FRDC*))
2497:                     ELSE
2498:                       IF STRGTYPE(GATTR.TYPTR) THEN
2499:                         BEGIN GENLDC(GATTR.TYPTR^.MAXLENG);
2500:                           GEN2(77(*CXP*),0(*SYS*),18(*FRDS*))
2501:                         END
2502:                       ELSE ERROR(125);
2503:               IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
2504:               TEST := SY <> COMMA;
2505:               IF NOT TEST THEN INSYMBOL
2506:             UNTIL TEST
2507:           END;
2508:         IF LKEY = 2 THEN
```

```
2509:          BEGIN LOADIDADDR(FILEPTR);
2510:            GEN2(77(*CXP*),0(*SYS*),21(*FRLN*));
2511:            IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2512:          END
2513:      END (*READ*) ;
2514:
2515:      PROCEDURE WRITE;
2516:        VAR LSP: STP; DEFAULT: BOOLEAN;
2517:            FILEPTR,LCP: CTP; LEN,LMIN,LMAX: INTEGER;
2518:      BEGIN FILEPTR := OUTPUTPTR;
2519:        IF (SY = IDENT) AND WASLPARENT THEN
2520:          BEGIN SEARCHID([FIELD,VARS,KONST,FUNC],LCP);
2521:            IF LCP^.IDTYPE <> NIL THEN
2522:              IF LCP^.IDTYPE^.FORM = FILES THEN
2523:                IF LCP^.IDTYPE^.FILTYPE = CHARPTR THEN
2524:                  BEGIN INSYMBOL; FILEPTR := LCP;
2525:                    IF NOT (SY IN [COMMA,RPARENT]) THEN ERROR(20);
2526:                    IF SY = COMMA THEN INSYMBOL
2527:                  END
2528:          END;
2529:        IF (SY IN FACBEGSYS) AND WASLPARENT THEN
2530:          BEGIN
2531:            REPEAT LOADIDADDR(FILEPTR);
2532:              EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
2533:              LSP := GATTR.TYPTR;
2534:              IF LSP <> NIL THEN
2535:                IF LSP^.FORM <= SUBRANGE THEN LOAD
2536:                ELSE LOADADDRESS;
2537:              IF SY = COLON THEN
2538:                BEGIN INSYMBOL;
2539:                  EXPRESSION(FSYS + [COMMA,COLON,RPARENT]);
2540:                  IF GATTR.TYPTR <> NIL THEN
2541:                    IF GATTR.TYPTR <> INTPTR THEN ERROR(20);
2542:                  LOAD; DEFAULT := FALSE
2543:                END
2544:              ELSE DEFAULT := TRUE;
2545:              IF LSP = INTPTR THEN
2546:                BEGIN IF DEFAULT THEN GENLDC(0);
2547:                  GEN2(77(*CXP*),0(*SYS*),13(*FWRI*))
2548:                END
2549:              ELSE
2550:                IF LSP = REALPTR THEN
2551:                  BEGIN IF DEFAULT THEN GENLDC(0);
2552:                    IF SY = COLON THEN
2553:                      BEGIN INSYMBOL;
2554:                        EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD;
2555:                        IF GATTR.TYPTR <> NIL THEN
2556:                          IF GATTR.TYPTR <> INTPTR THEN ERROR(125)
2557:                      END
2558:                    ELSE GENLDC(0);
2559:                    GEN2(77(*CXP*),0(*SYS*),15(*FWRR*))
2560:                  END
2561:                ELSE
2562:                  IF LSP = CHARPTR THEN
2563:                    BEGIN IF DEFAULT THEN GENLDC(0);
2564:                      GEN2(77(*CXP*),0(*SYS*),17(*FWRC*))
2565:                    END
2566:                  ELSE
2567:                    IF STRGTYPE(LSP) THEN
2568:                      BEGIN IF DEFAULT THEN GENLDC(0);
2569:                        GEN2(77(*CXP*),0(*SYS*),19(*FWRS*))
2570:                      END
2571:                    ELSE
2572:                      IF PAOFCHAR(LSP) THEN
2573:                        BEGIN LMAX := 0;
2574:                          IF LSP^.INXTYPE <> NIL THEN
```

```
2575:                           BEGIN GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
2576:                              LMAX := LMAX - LMIN + 1
2577:                           END;
2578:                         IF DEFAULT THEN GENLDC(LMAX);
2579:                         GENLDC(LMAX);
2580:                         GEN2(77(*CXP*),0(*SYS*),20(*FWRB*))
2581:                       END
2582:                     ELSE ERROR(125);
2583:               IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*));
2584:               TEST := SY <> COMMA;
2585:               IF NOT TEST THEN INSYMBOL
2586:             UNTIL TEST;
2587:           END;
2588:         IF LKEY = 4 THEN (*WRITELN*)
2589:           BEGIN LOADIDADDR(FILEPTR);
2590:             GEN2(77(*CXP*),0(*SYS*),22(*FWLN*));
2591:             IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2592:           END
2593:       END (*WRITE*) ;
2594:
2595:       PROCEDURE CALLNONSPECIAL;
2596:         VAR NXT,LCP: CTP; LSP: STP; LB: BOOLEAN;
2597:             LMIN,LMAX: INTEGER;
2598:       BEGIN
2599:         WITH FCP^ DO
2600:           BEGIN NXT := NEXT;
2601:             IF PFDECKIND = DECLARED THEN
2602:               IF PFKIND <> ACTUAL THEN ERROR(400)
2603:           END;
2604:         IF SY = LPARENT THEN
2605:           BEGIN
2606:             REPEAT
2607:               IF NXT = NIL THEN ERROR(126);
2608:               INSYMBOL;
2609:               EXPRESSION(FSYS + [COMMA,RPARENT]);
2610:               IF (GATTR.TYPTR <> NIL) AND (NXT <> NIL) THEN
2611:                 BEGIN LSP := NXT^.IDTYPE;
2612:                   IF LSP <> NIL THEN
2613:                     BEGIN
2614:                       IF NXT^.VKIND = ACTUAL THEN
2615:                         IF GATTR.TYPTR^.FORM <= POWER THEN
2616:                           BEGIN LB := (GATTR.TYPTR = CHARPTR)
2617:                                       AND (GATTR.KIND = CST);
2618:                             LOAD;
2619:                             IF LSP^.FORM = POWER THEN
2620:                               GEN1(32(*ADJ*),LSP^.SIZE)
2621:                             ELSE
2622:                             IF (LSP^.FORM = SUBRANGE)
2623:                                   AND RANGECHECK THEN
2624:                               BEGIN GENLDC(LSP^.MIN.IVAL);
2625:                                 GENLDC(LSP^.MAX.IVAL);
2626:                                 GEN0(8(*CHK*))
2627:                               END
2628:                             ELSE
2629:                             IF (GATTR.TYPTR = INTPTR) AND
2630:                                   COMPTYPES(LSP,REALPTR) THEN
2631:                               BEGIN GEN0(10(*FLT*));
2632:                                 GATTR.TYPTR := REALPTR
2633:                               END
2634:                             ELSE
2635:                             IF LB AND STRGTYPE(LSP) THEN
2636:                               GATTR.TYPTR := STRGPTR
2637:                           END
2638:                         ELSE (*FORM > POWER*)
2639:                           BEGIN LB := STRGTYPE(GATTR.TYPTR)
2640:                                       AND (GATTR.KIND = CST);
```

```
2641:                           LOADADDRESS;
2642:                           IF LB AND PAOFCHAR(LSP) THEN
2643:                             IF NOT LSP^.AISSTRNG THEN
2644:                               BEGIN GEN0(80(*S1P*));
2645:                                 IF LSP^.INXTYPE <> NIL THEN
2646:                                   BEGIN
2647:                                     GETBOUNDS(LSP^.INXTYPE,LMIN,LMAX);
2648:                                     IF LMAX-LMIN+1 <>
2649:                                         GATTR.TYPTR^.MAXLENG THEN ERROR(142);
2650:                                   END;
2651:                                 GATTR.TYPTR := LSP
2652:                               END
2653:                         END
2654:                       ELSE (*VKIND = FORMAL*)
2655:                         IF GATTR.KIND = VARBL THEN
2656:                           BEGIN LOADADDRESS;
2657:                             IF (LSP^.FORM=POWER) THEN
2658:                               IF GATTR.TYPTR^.SIZE <>
2659:                                   LSP^.SIZE THEN ERROR(142)
2660:                           END
2661:                         ELSE ERROR(154);
2662:                       IF NOT COMPTYPES(LSP,GATTR.TYPTR) THEN ERROR(142)
2663:                     END
2664:                 END;
2665:               IF NXT <> NIL THEN NXT := NXT^.NEXT
2666:             UNTIL SY <> COMMA;
2667:             IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
2668:           END (*LPARENT*) ;
2669:         IF NXT <> NIL THEN ERROR(126);
2670:         WITH FCP^ DO
2671:           IF PFDECKIND = DECLARED THEN
2672:             BEGIN
2673:               IF KLASS = FUNC THEN
2674:                 BEGIN GENLDC(0); GENLDC(0) END;
2675:               IF PFSEG <> SEG THEN GEN2(77(*CXP*),PFSEG,PFNAME)
2676:               ELSE
2677:                 IF PFLEV = 0 THEN GEN1(66(*CBP*),PFNAME)
2678:                 ELSE
2679:                   IF PFLEV = LEVEL THEN GEN1(78(*CLP*),PFNAME)
2680:                   ELSE
2681:                   IF PFLEV = 1 THEN GEN1(79(*CGP*),PFNAME)
2682:                   ELSE GEN1(46(*CIP*),PFNAME)
2683:             END
2684:           ELSE
2685:             IF (CSPNUM <> 21) AND (CSPNUM <> 22) THEN
2686:               GEN1(30(*CSP*),CSPNUM);
2687:         GATTR.TYPTR := FCP^.IDTYPE
2688:       END (*CALLNONSPECIAL*) ;
2689:
2690:     BEGIN (*CALL*)
2691:       IF FCP^.PFDECKIND = SPECIAL THEN
2692:         BEGIN WASLPARENT := TRUE; LKEY := FCP^.KEY;
2693:           IF SY = LPARENT THEN INSYMBOL
2694:           ELSE
2695:             IF LKEY IN [2,4,5,6] THEN WASLPARENT := FALSE
2696:             ELSE ERROR(9);
2697:           IF LKEY IN [7,8,9,10,11,13,14,25,36] THEN
2698:             BEGIN EXPRESSION(FSYS + [COMMA,RPARENT]); LOAD END;
2699:           CASE LKEY OF
2700:              1,2: READ;
2701:              3,4: WRITE;
2702:              5,6: BEGIN (*EOF & EOLN*)
2703:                     IF WASLPARENT THEN
2704:                       BEGIN VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
2705:                         IF GATTR.TYPTR <> NIL THEN
2706:                           IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125)
```

```
2707:                         ELSE
2708:                           IF (GATTR.TYPTR^.FILTYPE <> CHARPTR) AND
2709:                               (LKEY = 6) THEN ERROR(399)
2710:                     END
2711:                   ELSE
2712:                     LOADIDADDR(INPUTPTR);
2713:                   GENLDC(0); GENLDC(0);
2714:                   IF LKEY = 5 THEN GEN2(77(*CXP*),0(*SYS*),10(*FEOF*))
2715:                   ELSE GEN2(77(*CXP*),0(*SYS*),11(*FEOLN*));
2716:                   GATTR.TYPTR := BOOLPTR
2717:                 END (*EOF*) ;
2718:             7,8: BEGIN GENLDC(1); (*PREDSUCC*)
2719:                   IF GATTR.TYPTR <> NIL THEN
2720:                     IF GATTR.TYPTR^.FORM = SCALAR THEN
2721:                       IF LKEY = 8 THEN GEN0(2(*ADI*))
2722:                       ELSE GEN0(21(*SBI*))
2723:                     ELSE ERROR(115)
2724:                 END (*PREDSUCC*) ;
2725:               9: BEGIN (*ORD*)
2726:                   IF GATTR.TYPTR <> NIL THEN
2727:                     IF GATTR.TYPTR^.FORM >= POWER THEN ERROR(125);
2728:                   GATTR.TYPTR := INTPTR
2729:                 END (*ORD*) ;
2730:              10: BEGIN (*SQR*)
2731:                   IF GATTR.TYPTR <> NIL THEN
2732:                     IF GATTR.TYPTR = INTPTR THEN GEN0(24(*SQI*))
2733:                     ELSE
2734:                       IF GATTR.TYPTR = REALPTR THEN GEN0(25(*SQR*))
2735:                       ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
2736:                 END (*SQR*) ;
2737:              11: BEGIN (*ABS*)
2738:                   IF GATTR.TYPTR <> NIL THEN
2739:                     IF GATTR.TYPTR = INTPTR THEN GEN0(0(*ABI*))
2740:                     ELSE
2741:                       IF GATTR.TYPTR = REALPTR THEN GEN0(1(*ABR*))
2742:                       ELSE BEGIN ERROR(125); GATTR.TYPTR := INTPTR END
2743:                 END (*ABS*) ;
2744:              12: NEWSTMT;
2745:           13,14: UNITIO;
2746:              15: CONCAT;
2747:              16: BEGIN (*LENGTH*)
2748:                   STRGVAR(FSYS + [RPARENT],FALSE);
2749:                   GEN0(62(*LDB*)); GATTR.TYPTR := INTPTR
2750:                 END (*LENGTH*) ;
2751:              17: BEGIN (*INSERT*)
2752:                   STRGVAR(FSYS + [COMMA],FALSE);
2753:                   IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2754:                   STRGVAR(FSYS + [COMMA],TRUE);
2755:                   GENLDC(GATTR.TYPTR^.MAXLENG);
2756:                   IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2757:                   EXPRESSION(FSYS + [RPARENT]); LOAD;
2758:                   IF GATTR.TYPTR <> NIL THEN
2759:                     IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2760:                   GEN2(77(*CXP*),0(*SYS*),24(*SINSERT*))
2761:                 END (*INSERT*) ;
2762:         43,18,19: COPYDELETE;
2763:              20: BEGIN (*POS*)
2764:                   STRGVAR(FSYS + [COMMA],FALSE);
2765:                   IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2766:                   STRGVAR(FSYS + [RPARENT],FALSE);
2767:                   GENLDC(0); GENLDC(0);
2768:                   GEN2(77(*CXP*),0(*SYS*),27(*SPOS*));
2769:                   GATTR.TYPTR := INTPTR
2770:                 END (*POS*) ;
2771:         27,21,22: MOVE;
2772:              23: EXIT;
```

```
2773:              24: BEGIN (*IDSEARCH*)
2774:                    VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2775:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2776:                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
2777:                    GEN1(30(*CSP*),7(*IDS*))
2778:                  END (*IDSEARCH*) ;
2779:              25: BEGIN (*TREESEARCH*)
2780:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2781:                    VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2782:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2783:                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
2784:                    GATTR.TYPTR := INTPTR;
2785:                    GEN1(30(*CSP*),8(*TRS*))
2786:                  END (*TREESEARCH*) ;
2787:              26: BEGIN (*TIME*)
2788:                    VARIABLE(FSYS + [COMMA]); LOADADDRESS;
2789:                    IF GATTR.TYPTR <> NIL THEN
2790:                      IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2791:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2792:                    VARIABLE(FSYS + [RPARENT]); LOADADDRESS;
2793:                    IF GATTR.TYPTR <> NIL THEN
2794:                      IF GATTR.TYPTR <> INTPTR THEN ERROR(125);
2795:                    GEN1(30(*CSP*),9(*TIM*))
2796:                  END (*TIME*) ;
2797:       28,29,30: BEGIN (*OPEN*)
2798:                    VARIABLE(FSYS + [COMMA,RPARENT]); LOADADDRESS;
2799:                    IF GATTR.TYPTR <> NIL THEN
2800:                      IF GATTR.TYPTR^.FORM <> FILES THEN ERROR(125);
2801:                    IF SY = COMMA THEN INSYMBOL ELSE ERROR(20);
2802:                    STRGVAR(FSYS + [RPARENT],FALSE);
2803:                    IF (LKEY = 28) THEN GENLDC(0)
2804:                    ELSE GENLDC(1);
2805:                    GENLDC(0); GEN2(77(*CXP*),0(*SYS*),5(*FOPEN*))
2806:                    IF IOCHECK THEN GEN1(30(*CSP*),0(*IOC*))
2807:                  END (*OPEN*) ;
2808:              31: CLOSE;
2809: 32,33,34,35,40: GETPUTETC;
2810:              36: SCAN;
2811:           37,38: BLOCKIO;
2812:           39,42: DRAWSTUFF;
2813:              41: SIZEOF
2814:         END (*SPECIAL CASES*) ;
2815:         IF WASLPARENT THEN
2816:           IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
2817:       END (*SPECIAL PROCEDURES AND FUNCTIONS*)
2818:     ELSE CALLNONSPECIAL
2819:   END (*CALL*) ;
2820:
2821: (*$I XCOMP:E.TEXT *)
2822:
2823:   PROCEDURE EXPRESSION;
2824:     VAR LATTR: ATTR; LOP: OPERATOR; TYPIND: INTEGER;
2825:         LSIZE: ADDRRANGE; LSTRING,GSTRING: BOOLEAN;
2826:         LMIN,LMAX: INTEGER;
2827:
2828:     PROCEDURE FLOATIT(VAR FSP: STP);
2829:     BEGIN
2830:       IF GATTR.TYPTR = INTPTR THEN
2831:         BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
2832:       IF FSP = INTPTR THEN
2833:         BEGIN GEN0(9(*FLO*)); FSP := REALPTR END
2834:     END (*FLOATIT*) ;
2835:
2836:     PROCEDURE SIMPLEEXPRESSION(FSYS: SETOFSYS);
2837:       VAR LATTR: ATTR; LOP: OPERATOR; SIGNED: BOOLEAN;
2838:
```

```
2839:        PROCEDURE TERM(FSYS: SETOFSYS);
2840:          VAR LATTR: ATTR; LOP: OPERATOR;
2841:
2842:          PROCEDURE FACTOR(FSYS: SETOFSYS);
2843:            VAR LCP: CTP; LVP: CSP; VARPART,ALLCONST: BOOLEAN;
2844:                LSP: STP; HIGHVAL,LOWVAL,LIC,LOP: INTEGER;
2845:                CSTPART: SET OF 0..127;
2846:          BEGIN
2847:            IF NOT (SY IN FACBEGSYS) THEN
2848:              BEGIN ERROR(58); SKIP(FSYS + FACBEGSYS);
2849:                GATTR.TYPTR := NIL
2850:              END;
2851:            WHILE SY IN FACBEGSYS DO
2852:              BEGIN
2853:                CASE SY OF
2854:          (*ID*)  IDENT:
2855:                    BEGIN SEARCHID([KONST,VARS,FIELD,FUNC],LCP); INSYMBOL;
2856:                      IF LCP^.KLASS = FUNC THEN
2857:                        BEGIN CALL(FSYS,LCP); GATTR.KIND := EXPR END
2858:                      ELSE
2859:                        IF LCP^.KLASS = KONST THEN
2860:                          WITH GATTR, LCP^ DO
2861:                            BEGIN TYPTR := IDTYPE; KIND := CST;
2862:                              CVAL := VALUES
2863:                            END
2864:                      ELSE SELECTOR(FSYS,LCP);
2865:                    IF GATTR.TYPTR <> NIL THEN
2866:                      WITH GATTR,TYPTR^ DO
2867:                        IF FORM = SUBRANGE THEN TYPTR := RANGETYPE
2868:                  END;
2869:          (*CST*) INTCONST:
2870:                    BEGIN
2871:                      WITH GATTR DO
2872:                        BEGIN TYPTR := INTPTR; KIND := CST;
2873:                          CVAL := VAL
2874:                        END;
2875:                      INSYMBOL
2876:                    END;
2877:                  REALCONST:
2878:                    BEGIN
2879:                      WITH GATTR DO
2880:                        BEGIN TYPTR := REALPTR; KIND := CST;
2881:                          CVAL := VAL
2882:                        END;
2883:                      INSYMBOL
2884:                    END;
2885:                  STRINGCONST:
2886:                    BEGIN
2887:                      WITH GATTR DO
2888:                        BEGIN
2889:                          IF LGTH = 1 THEN TYPTR := CHARPTR
2890:                          ELSE
2891:                            BEGIN NEW(LSP,ARRAYS,TRUE,TRUE);
2892:                              LSP^ := STRGPTR^;
2893:                              LSP^.MAXLENG := LGTH;
2894:                              TYPTR := LSP
2895:                            END;
2896:                          KIND := CST; CVAL := VAL
2897:                        END;
2898:                      INSYMBOL
2899:                    END;
2900:          (*(*)   LPARENT:
2901:                    BEGIN INSYMBOL; EXPRESSION(FSYS + [RPARENT]);
2902:                      IF SY = RPARENT THEN INSYMBOL ELSE ERROR(4)
2903:                    END;
2904:          (*NOT*) NOTSY:
```

```
2905:                   BEGIN INSYMBOL; FACTOR(FSYS);
2906:                     LOAD; GEN0(19(*NOT*));
2907:                     IF GATTR.TYPTR <> NIL THEN
2908:                       IF GATTR.TYPTR <> BOOLPTR THEN
2909:                         BEGIN ERROR(135); GATTR.TYPTR := NIL END;
2910:                   END;
2911:           (*[*)   LBRACK:
2912:                   BEGIN INSYMBOL; CSTPART := [ ]; VARPART := FALSE;
2913:                     NEW(LSP,POWER);
2914:                     WITH LSP^ DO
2915:                       BEGIN ELSET := NIL; SIZE := 0; FORM := POWER END;
2916:                     IF SY = RBRACK THEN
2917:                       BEGIN
2918:                         WITH GATTR DO
2919:                           BEGIN TYPTR := LSP; KIND := CST END;
2920:                         INSYMBOL
2921:                       END
2922:                     ELSE
2923:                       BEGIN
2924:                         REPEAT EXPRESSION(FSYS + [COMMA,RBRACK,COLON]);
2925:                           IF GATTR.TYPTR <> NIL THEN
2926:                             IF GATTR.TYPTR^.FORM <> SCALAR THEN
2927:                               BEGIN ERROR(136); GATTR.TYPTR := NIL END
2928:                             ELSE
2929:                               IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
2930:                                 BEGIN ALLCONST := FALSE; LOP := 23(*SGS*);
2931:                                   IF (GATTR.KIND = CST) AND
2932:                                      (GATTR.CVAL.IVAL <= 127) THEN
2933:                                     BEGIN ALLCONST := TRUE;
2934:                                       LOWVAL := GATTR.CVAL.IVAL;
2935:                                       HIGHVAL := LOWVAL
2936:                                     END;
2937:                                   LIC := IC; LOAD;
2938:                                   IF SY = COLON THEN
2939:                                     BEGIN INSYMBOL; LOP := 20(*SRS*);
2940:                                       EXPRESSION(FSYS + [COMMA,RBRACK]);
2941:                                       IF COMPTYPES(LSP^.ELSET,GATTR.TYPTR) THEN
2942:                                       ELSE
2943:                                         BEGIN ERROR(137); GATTR.TYPTR:=NIL END;
2944:                                       IF ALLCONST THEN
2945:                                         IF (GATTR.KIND = CST) AND
2946:                                            (GATTR.CVAL.IVAL <= 127) THEN
2947:                                             HIGHVAL := GATTR.CVAL.IVAL
2948:                                         ELSE
2949:                                           BEGIN LOAD; ALLCONST := FALSE END
2950:                                       ELSE LOAD
2951:                                     END;
2952:                                   IF ALLCONST THEN
2953:                                     BEGIN IC := LIC; (*FORGET FIRST CONST*)
2954:                                       CSTPART := CSTPART + [LOWVAL..HIGHVAL]
2955:                                     END
2956:                                   ELSE
2957:                                     BEGIN GEN0(LOP);
2958:                                       IF VARPART THEN GEN0(28(*UNI*))
2959:                                       ELSE VARPART := TRUE
2960:                                     END;
2961:                                   LSP^.ELSET := GATTR.TYPTR;
2962:                                   GATTR.TYPTR := LSP
2963:                                 END
2964:                               ELSE ERROR(137);
2965:                           TEST := SY <> COMMA;
2966:                           IF NOT TEST THEN INSYMBOL
2967:                         UNTIL TEST;
2968:                         IF SY = RBRACK THEN INSYMBOL ELSE ERROR(12)
2969:                       END;
2970:                     IF VARPART THEN
```

```
2971:                       BEGIN
2972:                         IF CSTPART <> [ ] THEN
2973:                           BEGIN
2974:                             SCONST^.PVAL := CSTPART;
2975:                             SCONST^.CCLASS := PSET;
2976:                             GATTR.CVAL.VALP := SCONST;
2977:                             GATTR.KIND := CST;
2978:                             LOAD; GEN0(28(*UNI*))
2979:                           END;
2980:                         GATTR.KIND := EXPR
2981:                       END
2982:                     ELSE
2983:                       BEGIN
2984:                         SCONST^.PVAL := CSTPART;
2985:                         SCONST^.CCLASS := PSET;
2986:                         GATTR.CVAL.VALP := SCONST;
2987:                         GATTR.KIND := CST
2988:                       END
2989:                   END
2990:               END (*CASE*) ;
2991:               IF NOT (SY IN FSYS) THEN
2992:                 BEGIN ERROR(6); SKIP(FSYS + FACBEGSYS) END
2993:             END (*WHILE*)
2994:         END (*FACTOR*) ;
2995:
2996:       BEGIN (*TERM*)
2997:         FACTOR(FSYS + [MULOP]);
2998:         WHILE SY = MULOP DO
2999:           BEGIN LOAD; LATTR := GATTR; LOP := OP;
3000:             INSYMBOL; FACTOR(FSYS + [MULOP]); LOAD;
3001:             IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3002:               CASE LOP OF
3003:       (***)     MUL:  IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR)
3004:                         THEN GEN0(15(*MPI*))
3005:                       ELSE
3006:                         BEGIN FLOATIT(LATTR.TYPTR);
3007:                           IF (LATTR.TYPTR = REALPTR) AND
3008:                              (GATTR.TYPTR = REALPTR) THEN GEN0(16(*MPR*))
3009:                           ELSE
3010:                             IF (LATTR.TYPTR^.FORM = POWER)
3011:                                 AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3012:                               GEN0(12(*INT*))
3013:                             ELSE BEGIN ERROR(134); GATTR.TYPTR:=NIL END
3014:                         END;
3015:       (*/*)     RDIV: BEGIN FLOATIT(LATTR.TYPTR);
3016:                         IF (LATTR.TYPTR = REALPTR) AND
3017:                            (GATTR.TYPTR = REALPTR) THEN GEN0(7(*DVR*))
3018:                         ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3019:                       END;
3020:       (*DIV*)   IDIV: IF (LATTR.TYPTR = INTPTR) AND
3021:                          (GATTR.TYPTR = INTPTR) THEN GEN0(6(*DVI*))
3022:                       ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END;
3023:       (*MOD*)   IMOD: IF (LATTR.TYPTR = INTPTR) AND
3024:                          (GATTR.TYPTR = INTPTR) THEN GEN0(14(*MOD*))
3025:                       ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END;
3026:       (*AND*)   ANDOP:IF (LATTR.TYPTR = BOOLPTR) AND
3027:                          (GATTR.TYPTR = BOOLPTR) THEN GEN0(4(*AND*))
3028:                       ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3029:               END (*CASE*)
3030:             ELSE GATTR.TYPTR := NIL
3031:           END (*WHILE*)
3032:       END (*TERM*) ;
3033:
3034:     BEGIN (*SIMPLEEXPRESSION*)
3035:       SIGNED := FALSE;
3036:       IF (SY = ADDOP) AND (OP IN [PLUS,MINUS]) THEN
```

```
3037:          BEGIN SIGNED := OP = MINUS; INSYMBOL END;
3038:        TERM(FSYS + [ADDOP]);
3039:        IF SIGNED THEN
3040:          BEGIN LOAD;
3041:            IF GATTR.TYPTR = INTPTR THEN GEN0(17(*NGI*))
3042:            ELSE
3043:              IF GATTR.TYPTR = REALPTR THEN GEN0(18(*NGR*))
3044:              ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3045:          END;
3046:        WHILE SY = ADDOP DO
3047:          BEGIN LOAD; LATTR := GATTR; LOP := OP;
3048:            INSYMBOL; TERM(FSYS + [ADDOP]); LOAD;
3049:            IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3050:              CASE LOP OF
3051:      (*+*)     PLUS:
3052:                  IF (LATTR.TYPTR = INTPTR)AND(GATTR.TYPTR = INTPTR) THEN
3053:                    GEN0(2(*ADI*))
3054:                  ELSE
3055:                    BEGIN FLOATIT(LATTR.TYPTR);
3056:                      IF (LATTR.TYPTR = REALPTR)AND(GATTR.TYPTR = REALPTR)
3057:                        THEN GEN0(3(*ADR*))
3058:                      ELSE IF (LATTR.TYPTR^.FORM = POWER)
3059:                               AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3060:                             GEN0(28(*UNI*))
3061:                           ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3062:                    END;
3063:      (*-*)     MINUS:
3064:                  IF (LATTR.TYPTR = INTPTR) AND (GATTR.TYPTR = INTPTR) THEN
3065:                    GEN0(21(*SBI*))
3066:                  ELSE
3067:                    BEGIN FLOATIT(LATTR.TYPTR);
3068:                      IF (LATTR.TYPTR = REALPTR) AND (GATTR.TYPTR = REALPTR)
3069:                        THEN GEN0(22(*SBR*))
3070:                      ELSE
3071:                        IF (LATTR.TYPTR^.FORM = POWER)
3072:                            AND COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3073:                          GEN0(5(*DIF*))
3074:                        ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3075:                    END;
3076:      (*OR*)    OROP:
3077:                  IF (LATTR.TYPTR = BOOLPTR) AND (GATTR.TYPTR = BOOLPTR) THEN
3078:                    GEN0(13(*IOR*))
3079:                  ELSE BEGIN ERROR(134); GATTR.TYPTR := NIL END
3080:              END (*CASE*)
3081:            ELSE GATTR.TYPTR := NIL
3082:          END (*WHILE*)
3083:      END (*SIMPLEEXPRESSION*) ;
3084:
3085:      PROCEDURE MAKEPA(VAR STRGFSP: STP; PAFSP: STP);
3086:        VAR LMIN,LMAX: INTEGER;
3087:      BEGIN
3088:        IF PAFSP^.INXTYPE <> NIL THEN
3089:          BEGIN GETBOUNDS(PAFSP^.INXTYPE,LMIN,LMAX);
3090:            IF LMAX-LMIN+1 <> STRGFSP^.MAXLENG THEN ERROR(129)
3091:          END;
3092:        STRGFSP := PAFSP
3093:      END (*MAKEPA*) ;
3094:
3095:    BEGIN (*EXPRESSION*)
3096:      SIMPLEEXPRESSION(FSYS + [RELOP]);
3097:      IF SY = RELOP THEN
3098:        BEGIN
3099:          LSTRING := (STRGTYPE(GATTR.TYPTR) AND (GATTR.KIND = CST);
3100:          IF GATTR.TYPTR <> NIL THEN
3101:            IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3102:            ELSE LOADADDRESS;
```

```
3103:          LATTR := GATTR; LOP := OP;
3104:          INSYMBOL; SIMPLEEXPRESSION(FSYS);
3105:          GSTRING := STRGTYPE(GATTR.TYPTR) AND (GATTR.KIND = CST);
3106:          IF GATTR.TYPTR <> NIL THEN
3107:            IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3108:            ELSE LOADADDRESS;
3109:          IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3110:            IF LOP = INOP THEN
3111:              IF GATTR.TYPTR^.FORM = POWER THEN
3112:                IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR^.ELSET) THEN
3113:                  GEN0(11(*INN*))
3114:                ELSE BEGIN ERROR(129); GATTR.TYPTR := NIL END
3115:              ELSE BEGIN ERROR(130); GATTR.TYPTR := NIL END
3116:            ELSE
3117:              BEGIN
3118:                IF LATTR.TYPTR <> GATTR.TYPTR THEN FLOATIT(LATTR.TYPTR);
3119:                IF LSTRING THEN
3120:                  BEGIN
3121:                    IF PAOFCHAR(GATTR.TYPTR) THEN
3122:                      IF NOT GATTR.TYPTR^.AISSTRNG THEN
3123:                        BEGIN GEN0(29(*S2P*));
3124:                          MAKEPA(LATTR.TYPTR,GATTR.TYPTR)
3125:                        END
3126:                  END
3127:                ELSE
3128:                  IF GSTRING THEN
3129:                    BEGIN
3130:                      IF PAOFCHAR(LATTR.TYPTR) THEN
3131:                        IF NOT LATTR.TYPTR^.AISSTRNG THEN
3132:                          BEGIN GEN0(80(*S1P*));
3133:                            MAKEPA(GATTR.TYPTR,LATTR.TYPTR)
3134:                          END;
3135:                    END;
3136:                IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3137:                  BEGIN LSIZE := LATTR.TYPTR^.SIZE;
3138:                    CASE LATTR.TYPTR^.FORM OF
3139:                      SCALAR:
3140:                        IF LATTR.TYPTR = REALPTR THEN TYPIND := 1
3141:                        ELSE
3142:                          IF LATTR.TYPTR = BOOLPTR THEN TYPIND := 3
3143:                          ELSE TYPIND := 0;
3144:                      POINTER:
3145:                        BEGIN
3146:                          IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3147:                          TYPIND := 0
3148:                        END;
3149:                      POWER:
3150:                        BEGIN
3151:                          IF LOP IN [LTOP,GTOP] THEN ERROR(132);
3152:                          TYPIND := 4
3153:                        END;
3154:                      ARRAYS:
3155:                        BEGIN
3156:                          TYPIND := 6;
3157:                          IF PAOFCHAR(LATTR.TYPTR) THEN
3158:                            IF LATTR.TYPTR^.AISSTRNG THEN TYPIND := 2
3159:                            ELSE
3160:                              BEGIN TYPIND := 5;
3161:                                IF LATTR.TYPTR^.INXTYPE <> NIL THEN
3162:                                  BEGIN
3163:                                    GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3164:                                    LSIZE := LMAX - LMIN + 1
3165:                                  END
3166:                              END
3167:                          ELSE
3168:                            IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131)
```

```
3169:                       END;
3170:                     RECORDS:
3171:                       BEGIN
3172:                         IF LOP IN [LTOP,LEOP,GTOP,GEOP] THEN ERROR(131);
3173:                         TYPIND := 6
3174:                       END;
3175:                     FILES:
3176:                       BEGIN ERROR(133); TYPIND := 0 END
3177:                   END;
3178:                   CASE LOP OF
3179:                     LTOP: GEN2(53(*LES*),TYPIND,LSIZE);
3180:                     LEOP: GEN2(52(*LEQ*),TYPIND,LSIZE);
3181:                     GTOP: GEN2(49(*GRT*),TYPIND,LSIZE);
3182:                     GEOP: GEN2(48(*GEQ*),TYPIND,LSIZE);
3183:                     NEOP: GEN2(55(*NEQ*),TYPIND,LSIZE);
3184:                     EQOP: GEN2(47(*EQU*),TYPIND,LSIZE)
3185:                   END
3186:                 END
3187:               ELSE ERROR(129)
3188:             END;
3189:         GATTR.TYPTR := BOOLPTR; GATTR.KIND := EXPR
3190:       END (*SY = RELOP*)
3191:   END (*EXPRESSION*) ;
3192:
3193:   PROCEDURE STATEMENT(FSYS: SETOFSYS);
3194:     LABEL 1;
3195:     VAR LCP: CTP; TTOP: DISPRANGE; LLP: LABELP; HEAP: ^INTEGER;
3196:
3197:     PROCEDURE ASSIGNMENT(FCP: CTP);
3198:       VAR LATTR: ATTR; CSTRING,PAONLEFT: BOOLEAN; LMIN,LMAX: INTEGER;
3199:     BEGIN SELECTOR(FSYS + [BECOMES],FCP);
3200:       IF SY = BECOMES THEN
3201:         BEGIN LMAX := 0; CSTRING := FALSE;
3202:           IF GATTR.TYPTR <> NIL THEN
3203:             IF (GATTR.ACCESS = INDRCT) OR (GATTR.TYPTR^.FORM > POWER) THEN
3204:               LOADADDRESS;
3205:           PAONLEFT := PAOFCHAR(GATTR.TYPTR);
3206:           LATTR := GATTR;
3207:           INSYMBOL; EXPRESSION(FSYS);
3208:           IF GATTR.KIND = CST THEN
3209:             CSTRING := (GATTR.TYPTR = CHARPTR) OR STRGTYPE(GATTR.TYPTR);
3210:           IF GATTR.TYPTR <> NIL THEN
3211:             IF GATTR.TYPTR^.FORM <= POWER THEN LOAD
3212:             ELSE LOADADDRESS;
3213:           IF (LATTR.TYPTR <> NIL) AND (GATTR.TYPTR <> NIL) THEN
3214:             BEGIN
3215:               IF GATTR.TYPTR = INTPTR THEN
3216:                 IF COMPTYPES(REALPTR,LATTR.TYPTR) THEN
3217:                   BEGIN GEN0(10(*FLT*)); GATTR.TYPTR := REALPTR END;
3218:               IF PAONLEFT THEN
3219:                 IF LATTR.TYPTR^.AISSTRNG THEN
3220:                   IF CSTRING AND (GATTR.TYPTR = CHARPTR) THEN
3221:                     GATTR.TYPTR := STRGPTR
3222:                   ELSE
3223:                 ELSE
3224:                   IF LATTR.TYPTR^.INXTYPE <> NIL THEN
3225:                     BEGIN GETBOUNDS(LATTR.TYPTR^.INXTYPE,LMIN,LMAX);
3226:                       LMAX := LMAX - LMIN + 1;
3227:                       IF CSTRING AND (GATTR.TYPTR <> CHARPTR) THEN
3228:                         BEGIN GEN0(80(*S1P*));
3229:                           IF LMAX <> GATTR.TYPTR^.MAXLENG THEN ERROR(129);
3230:                           GATTR.TYPTR := LATTR.TYPTR
3231:                         END
3232:                     END
3233:                   ELSE GATTR.TYPTR := LATTR.TYPTR;
3234:               IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
```

```
3235:                 CASE LATTR.TYPTR^.FORM OF
3236:                   SUBRANGE: BEGIN
3237:                               IF RANGECHECK THEN
3238:                                 BEGIN
3239:                                   GENLDC(LATTR.TYPTR^.MIN.IVAL);
3240:                                   GENLDC(LATTR.TYPTR^.MAX.IVAL);
3241:                                   GEN0(8(*CHK*))
3242:                                 END;
3243:                               STORE(LATTR)
3244:                             END;
3245:                   POWER:    BEGIN
3246:                               GEN1(32(*ADJ*),LATTR.TYPTR^.SIZE);
3247:                               STORE(LATTR)
3248:                             END;
3249:                   SCALAR,
3250:                   POINTER: STORE(LATTR);
3251:                   ARRAYS:  IF PAONLEFT THEN
3252:                              IF LATTR.TYPTR^.AISSTRNG THEN
3253:                                GEN1(42(*SAS*),LATTR.TYPTR^.MAXLENG)
3254:                              ELSE GEN1(41(*MVB*),LMAX)
3255:                            ELSE GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3256:                   RECORDS: GEN1(40(*MOV*),LATTR.TYPTR^.SIZE);
3257:                   FILES:   ERROR(146)
3258:                 END
3259:               ELSE ERROR(129)
3260:             END
3261:         END (*SY = BECOMES*)
3262:       ELSE ERROR(51)
3263:     END (*ASSIGNMENT*) ;
3264:
3265:     PROCEDURE GOTOSTATEMENT;
3266:       VAR LLP: LABELP; FOUND: BOOLEAN; TTOP: DISPRANGE;
3267:     BEGIN
3268:       IF NOT GOTOOK THEN ERROR(6);
3269:       IF SY = INTCONST THEN
3270:         BEGIN
3271:           FOUND := FALSE; TTOP := TOP;
3272:           WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP - 1;
3273:           LLP := DISPLAY[TTOP].FLABEL;
3274:           WHILE (LLP <> NIL) AND NOT FOUND DO
3275:             WITH LLP^ DO
3276:               IF LABVAL = VAL.IVAL THEN
3277:                 BEGIN FOUND := TRUE;
3278:                   GENJMP(57(*UJP*),CODELBP)
3279:                 END
3280:               ELSE LLP := NEXTLAB;
3281:           IF NOT FOUND THEN ERROR(167);
3282:           INSYMBOL
3283:         END
3284:       ELSE ERROR(15)
3285:     END (*GOTOSTATEMENT*) ;
3286:
3287:     PROCEDURE COMPOUNDSTATEMENT;
3288:     BEGIN
3289:       REPEAT
3290:         REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
3291:         UNTIL NOT (SY IN STATBEGSYS);
3292:         TEST := SY <> SEMICOLON;
3293:         IF NOT TEST THEN INSYMBOL
3294:       UNTIL TEST;
3295:       IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
3296:     END (*COMPOUNDSTATEMENET*) ;
3297:
3298:     PROCEDURE IFSTATEMENT;
3299:       VAR LCIX1,LCIX2: LBP;
3300:     BEGIN EXPRESSION(FSYS + [THENSY]);
```

```
3301:         GENLABEL(LCIX1); GENFJP(LCIX1);
3302:         IF SY = THENSY THEN INSYMBOL ELSE ERROR(52);
3303:         STATEMENT(FSYS + [ELSESY]);
3304:         IF SY = ELSESY THEN
3305:           BEGIN GENLABEL(LCIX2); GENJMP(57(*UJP*),LCIX2);
3306:             PUTLABEL(LCIX1);
3307:             INSYMBOL; STATEMENT(FSYS);
3308:             PUTLABEL(LCIX2)
3309:           END
3310:         ELSE PUTLABEL(LCIX1)
3311:       END (*IFSTATEMENT*) ;
3312:
3313:       PROCEDURE CASESTATEMENT;
3314:         LABEL 1;
3315:         TYPE CIP = ^CASEINFO;
3316:              CASEINFO = RECORD
3317:                           NEXT: CIP;
3318:                           CSSTART: INTEGER;
3319:                           CSLAB: INTEGER
3320:                         END;
3321:         VAR LSP,LSP1: STP; FSTPTR,LPT1,LPT2,LPT3: CIP; LVAL: VALU;
3322:             LADDR, LCIX: LBP; NULSTMT, LMIN, LMAX: INTEGER;
3323:       BEGIN EXPRESSION(FSYS + [OFSY,COMMA,COLON]);
3324:         LOAD; GENLABEL(LCIX); GENJMP(57(*UJP*),LCIX);
3325:         LSP := GATTR.TYPTR;
3326:         IF LSP <> NIL THEN
3327:           IF (LSP^.FORM <> SCALAR) OR (LSP = REALPTR) THEN
3328:             BEGIN ERROR(144); LSP := NIL END;
3329:         IF SY = OFSY THEN INSYMBOL ELSE ERROR(8);
3330:         FSTPTR := NIL; GENLABEL(LADDR);
3331:         REPEAT
3332:           LPT3 := NIL;
3333:           REPEAT CONSTANT(FSYS + [COMMA,COLON],LSP1,LVAL);
3334:             IF LSP <> NIL THEN
3335:               IF COMPTYPES(LSP,LSP1) THEN
3336:                 BEGIN LPT1 := FSTPTR; LPT2 := NIL;
3337:                   WHILE LPT1 <> NIL DO
3338:                     WITH LPT1^ DO
3339:                       BEGIN
3340:                         IF CSLAB <= LVAL.IVAL THEN
3341:                           BEGIN IF CSLAB = LVAL.IVAL THEN ERROR(156);
3342:                             GOTO 1
3343:                           END;
3344:                         LPT2 := LPT1; LPT1 := NEXT
3345:                       END;
3346:       1:            NEW(LPT3);
3347:                     WITH LPT3^ DO
3348:                       BEGIN NEXT := LPT1; CSLAB := LVAL.IVAL;
3349:                         CSSTART := IC
3350:                       END;
3351:                     IF LPT2 = NIL THEN FSTPTR := LPT3
3352:                     ELSE LPT2^.NEXT := LPT3
3353:                 END
3354:               ELSE ERROR(147);
3355:             TEST := SY <> COMMA;
3356:             IF NOT TEST THEN INSYMBOL
3357:           UNTIL TEST;
3358:           IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
3359:           REPEAT STATEMENT(FSYS + [SEMICOLON])
3360:           UNTIL NOT (SY IN STATBEGSYS);
3361:           IF LPT3 <> NIL THEN
3362:             GENJMP(57(*UJP*),LADDR);
3363:           TEST := SY <> SEMICOLON;
3364:           IF NOT TEST THEN INSYMBOL
3365:         UNTIL TEST OR (SY = ENDSY);
3366:         PUTLABEL(LCIX);
```

```
3367:        IF FSTPTR <> NIL THEN
3368:          BEGIN LMAX := FSTPTR^.CSLAB;
3369:            LPT1 := FSTPTR; FSTPTR := NIL;
3370:            REPEAT LPT2 := LPT1^.NEXT; LPT1^.NEXT := FSTPTR;
3371:              FSTPTR := LPT1; LPT1 := LPT2
3372:            UNTIL LPT1 = NIL;
3373:            LMIN := FSTPTR^.CSLAB;
3374:                GEN0(44(*XJP*));
3375:                GENWORD(LMIN); GENWORD(LMAX);
3376:                NULSTMT := IC;
3377:                GENJMP(57(*UJP*),LADDR);
3378:                REPEAT
3379:                  WITH FSTPTR^ DO
3380:                    BEGIN
3381:                      WHILE CSLAB > LMIN DO
3382:                        BEGIN GENWORD(IC-NULSTMT); LMIN := LMIN + 1 END;
3383:                      GENWORD(IC-CSSTART);
3384:                      FSTPTR := NEXT; LMIN := LMIN + 1
3385:                    END
3386:                UNTIL FSTPTR = NIL;
3387:                PUTLABEL(LADDR)
3388:          END;
3389:          IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13)
3390:      END (*CASESTATEMENT*) ;
3391:
3392:      PROCEDURE REPEATSTATEMENT;
3393:        VAR LADDR: LBP;
3394:      BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
3395:        REPEAT
3396:          REPEAT STATEMENT(FSYS + [SEMICOLON,UNTILSY])
3397:          UNTIL NOT (SY IN STATBEGSYS);
3398:          TEST := SY <> SEMICOLON;
3399:          IF NOT TEST THEN INSYMBOL
3400:        UNTIL TEST;
3401:        IF SY = UNTILSY THEN
3402:          BEGIN INSYMBOL; EXPRESSION(FSYS); GENFJP(LADDR)
3403:          END
3404:        ELSE ERROR(53)
3405:      END (*REPEATSTATEMENT*) ;
3406:
3407:      PROCEDURE WHILESTATEMENT;
3408:        VAR LADDR, LCIX: LBP;
3409:      BEGIN GENLABEL(LADDR); PUTLABEL(LADDR);
3410:        EXPRESSION(FSYS + [DOSY]); GENLABEL(LCIX); GENFJP(LCIX);
3411:        IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
3412:        STATEMENT(FSYS); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX)
3413:      END (*WHILESTATEMENT*) ;
3414:
3415:      PROCEDURE FORSTATEMENT;
3416:        VAR LATTR: ATTR; LSP: STP;  LSY: SYMBOL;
3417:            LCIX, LADDR: LBP;
3418:      BEGIN
3419:        IF SY = IDENT THEN
3420:          BEGIN SEARCHID([VARS],LCP);
3421:            WITH LCP^, LATTR DO
3422:              BEGIN TYPTR := IDTYPE; KIND := VARBL;
3423:                IF VKIND = ACTUAL THEN
3424:                  BEGIN ACCESS := DRCT; VLEVEL := VLEV;
3425:                    DPLMT := VADDR
3426:                  END
3427:                ELSE BEGIN ERROR(155); TYPTR := NIL END
3428:              END;
3429:            IF LATTR.TYPTR <> NIL THEN
3430:              IF (LATTR.TYPTR^.FORM > SUBRANGE)
3431:                 OR COMPTYPES(REALPTR,LATTR.TYPTR) THEN
3432:                BEGIN ERROR(143); LATTR.TYPTR := NIL END;
```

```
3433:           INSYMBOL
3434:         END
3435:       ELSE
3436:         BEGIN ERROR(2); SKIP(FSYS + [BECOMES,TOSY,DOWNTOSY,DOSY])
3437:         END;
3438:       IF SY = BECOMES THEN
3439:         BEGIN INSYMBOL; EXPRESSION(FSYS + [TOSY,DOWNTOSY,DOSY]);
3440:           IF GATTR.TYPTR <> NIL THEN
3441:             IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
3442:               ELSE
3443:                 IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3444:                   BEGIN LOAD;
3445:                     IF LATTR.TYPTR <> NIL THEN
3446:                       IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK THEN
3447:                         BEGIN
3448:                           GENLDC(LATTR.TYPTR^.MIN.IVAL);
3449:                           GENLDC(LATTR.TYPTR^.MAX.IVAL);
3450:                           GEN0(8(*CHK*))
3451:                         END;
3452:                     STORE(LATTR)
3453:                   END
3454:                 ELSE ERROR(145)
3455:         END
3456:       ELSE
3457:         BEGIN ERROR(51); SKIP(FSYS + [TOSY,DOWNTOSY,DOSY]) END;
3458:       GENLABEL(LADDR);
3459:       IF SY IN [TOSY,DOWNTOSY] THEN
3460:         BEGIN LSY := SY; INSYMBOL; EXPRESSION(FSYS + [DOSY]);
3461:           IF GATTR.TYPTR <> NIL THEN
3462:             IF GATTR.TYPTR^.FORM <> SCALAR THEN ERROR(144)
3463:             ELSE
3464:               IF COMPTYPES(LATTR.TYPTR,GATTR.TYPTR) THEN
3465:                 BEGIN LOAD;
3466:                   IF LATTR.TYPTR <> NIL THEN
3467:                     IF (LATTR.TYPTR^.FORM = SUBRANGE) AND RANGECHECK THEN
3468:                       BEGIN
3469:                         GENLDC(LATTR.TYPTR^.MIN.IVAL);
3470:                         GENLDC(LATTR.TYPTR^.MAX.IVAL);
3471:                         GEN0(8(*CHK*))
3472:                       END;
3473:                   GEN2(56(*STR*),0,LC); PUTLABEL(LADDR);
3474:                   GATTR := LATTR; LOAD; GEN2(54(*LOD*),0,LC);
3475:                   LC := LC + INTSIZE;
3476:                   IF LC > LCMAX THEN LCMAX := LC;
3477:                   IF LSY = TOSY THEN GEN2(52(*LEQ*),0,INTSIZE)
3478:                   ELSE GEN2(48(*GEQ*),0,INTSIZE);
3479:                 END
3480:               ELSE ERROR(145)
3481:         END
3482:       ELSE BEGIN ERROR(55); SKIP(FSYS + [DOSY]) END;
3483:       GENLABEL(LCIX); GENJMP(33(*FJP*),LCIX);
3484:       IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
3485:       STATEMENT(FSYS);
3486:       GATTR := LATTR; LOAD; GENLDC(1);
3487:       IF LSY = TOSY THEN GEN0(2(*ADI*)) ELSE GEN0(21(*SBI*));
3488:       STORE(LATTR); GENJMP(57(*UJP*),LADDR); PUTLABEL(LCIX);
3489:       LC := LC - INTSIZE
3490:     END (*FORSTATEMENT*) ;
3491:
3492:
3493:     PROCEDURE WITHSTATEMENT;
3494:       VAR LCP: CTP; LCNT1,LCNT2: DISPRANGE;
3495:     BEGIN LCNT1 := 0; LCNT2 := 0;
3496:       REPEAT
3497:         IF SY = IDENT THEN
3498:           BEGIN SEARCHID([VARS,FIELD],LCP); INSYMBOL END
```

```
3499:          ELSE BEGIN ERROR(2); LCP := UVARPTR END;
3500:          SELECTOR(FSYS + [COMMA,DOSY],LCP);
3501:          IF GATTR.TYPTR <> NIL THEN
3502:            IF GATTR.TYPTR^.FORM = RECORDS THEN
3503:              IF TOP < DISPLIMIT THEN
3504:                BEGIN TOP := TOP + 1; LCNT1 := LCNT1 + 1;
3505:                  WITH DISPLAY[TOP] DO
3506:                    BEGIN FNAME := GATTR.TYPTR^.FSTFLD END;
3507:                  IF GATTR.ACCESS = DRCT THEN
3508:                    WITH DISPLAY[TOP] DO
3509:                      BEGIN OCCUR := CREC; CLEV := GATTR.VLEVEL;
3510:                        CDSPL := GATTR.DPLMT
3511:                      END
3512:                  ELSE
3513:                    BEGIN LOADADDRESS; GEN2(56(*STR*),0,LC);
3514:                      WITH DISPLAY[TOP] DO
3515:                        BEGIN OCCUR := VREC; VDSPL := LC END;
3516:                      LC := LC + PTRSIZE; LCNT2 := LCNT2 + PTRSIZE;
3517:                      IF LC > LCMAX THEN LCMAX := LC
3518:                    END
3519:                END
3520:              ELSE ERROR(250)
3521:            ELSE ERROR(140);
3522:          TEST := SY <> COMMA;
3523:          IF NOT TEST THEN INSYMBOL
3524:        UNTIL TEST;
3525:        IF SY = DOSY THEN INSYMBOL ELSE ERROR(54);
3526:        STATEMENT(FSYS);
3527:        TOP := TOP - LCNT1; LC := LC - LCNT2;
3528:      END (*WITHSTATEMENT*) ;
3529:
3530:    BEGIN (*STATEMENT*)
3531:      IF SY = INTCONST THEN (*LABEL*)
3532:        BEGIN TTOP := TOP;
3533:          WHILE DISPLAY[TTOP].OCCUR <> BLCK DO TTOP := TTOP-1;
3534:          LLP := DISPLAY[TTOP].FLABEL;
3535:          WHILE LLP <> NIL DO
3536:            WITH LLP^ DO
3537:              IF LABVAL = VAL.IVAL THEN
3538:                BEGIN
3539:                  IF CODELBP^.DEFINED THEN ERROR(165);
3540:                  PUTLABEL(CODELBP); GOTO 1
3541:                END
3542:              ELSE LLP := NEXTLAB;
3543:          ERROR(167);
3544:    1:    INSYMBOL;
3545:          IF SY = COLON THEN INSYMBOL ELSE ERROR(5)
3546:        END;
3547:      IF NOT (SY IN FSYS + [IDENT]) THEN
3548:        BEGIN ERROR(6); SKIP(FSYS) END;
3549:      IF SY IN STATBEGSYS + [IDENT] THEN
3550:        BEGIN MARK(HEAP); (*FOR LABEL CLEANUP*)
3551:          CASE SY OF
3552:            IDENT:    BEGIN SEARCHID([VARS,FIELD,FUNC,PROC],LCP);
3553:                        INSYMBOL;
3554:                        IF LCP^.KLASS = PROC THEN CALL(FSYS,LCP)
3555:                        ELSE ASSIGNMENT(LCP)
3556:                      END;
3557:            BEGINSY:  BEGIN INSYMBOL; COMPOUNDSTATEMENT END;
3558:            GOTOSY:   BEGIN INSYMBOL; GOTOSTATEMENT END;
3559:            IFSY:     BEGIN INSYMBOL; IFSTATEMENT END;
3560:            CASESY:   BEGIN INSYMBOL; CASESTATEMENT END;
3561:            WHILESY:  BEGIN INSYMBOL; WHILESTATEMENT END;
3562:            REPEATSY: BEGIN INSYMBOL; REPEATSTATEMENT END;
3563:            FORSY:    BEGIN INSYMBOL; FORSTATEMENT END;
3564:            WITHSY:   BEGIN INSYMBOL; WITHSTATEMENT END
```

```
3565:          END;
3566:          RELEASE(HEAP);
3567:          IF IC + 100 > MAXCODE THEN
3568:            BEGIN ERROR(253); IC := 0 END;
3569:          IF NOT (SY IN [SEMICOLON,ENDSY,ELSESY,UNTILSY]) THEN
3570:            BEGIN ERROR(6); SKIP(FSYS) END
3571:        END
3572:    END (*STATEMENT*) ;
3573:
3574:  (*$I XCOMP:F.TEXT *)
3575:
3576:    PROCEDURE BLOCK(FSYS: SETOFSYS; FSY: SYMBOL; FPROCP: CTP);
3577:      VAR LSY: SYMBOL;
3578:
3579:      PROCEDURE LABELDECLARATION;
3580:        VAR LLP: LABELP; REDEF: BOOLEAN;
3581:      BEGIN
3582:        REPEAT
3583:          IF SY = INTCONST THEN
3584:            WITH DISPLAY[TOP] DO
3585:              BEGIN LLP := FLABEL; REDEF := FALSE;
3586:                WHILE (LLP <> NIL) AND NOT REDEF DO
3587:                  IF LLP^.LABVAL <> VAL.IVAL THEN
3588:                    LLP := LLP^.NEXTLAB
3589:                  ELSE BEGIN REDEF := TRUE; ERROR(166) END;
3590:                IF NOT REDEF THEN
3591:                  BEGIN NEW(LLP);
3592:                    WITH LLP^ DO
3593:                      BEGIN LABVAL := VAL.IVAL;
3594:                        CODELBP := NIL; NEXTLAB := FLABEL
3595:                      END;
3596:                    FLABEL := LLP
3597:                  END;
3598:                INSYMBOL
3599:              END
3600:          ELSE ERROR(15);
3601:          IF NOT ( SY IN FSYS + [COMMA, SEMICOLON] ) THEN
3602:            BEGIN ERROR(6); SKIP(FSYS+[COMMA,SEMICOLON]) END;
3603:          TEST := SY <> COMMA;
3604:          IF NOT TEST THEN INSYMBOL
3605:        UNTIL TEST;
3606:        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14)
3607:      END (* LABELDECLARATION *) ;
3608:
3609:      PROCEDURE CONSTDECLARATION;
3610:        VAR LCP: CTP; LSP: STP; LVALU: VALU;
3611:      BEGIN
3612:        IF SY <> IDENT THEN
3613:          BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
3614:        WHILE SY = IDENT DO
3615:          BEGIN NEW(LCP,KONST);
3616:            WITH LCP^ DO
3617:              BEGIN NAME := ID; IDTYPE := NIL;
3618:                NEXT := NIL; KLASS := KONST
3619:              END;
3620:            INSYMBOL;
3621:            IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
3622:            CONSTANT(FSYS + [SEMICOLON],LSP,LVALU);
3623:            ENTERID(LCP);
3624:            LCP^.IDTYPE := LSP; LCP^.VALUES := LVALU;
3625:            IF SY = SEMICOLON THEN
3626:              BEGIN INSYMBOL;
3627:                IF NOT (SY IN FSYS + [IDENT]) THEN
3628:                  BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
3629:              END
3630:            ELSE ERROR(14)
```

```
3631:          END
3632:      END (*CONSTDECLARATION*) ;
3633:
3634:      PROCEDURE TYPEDECLARATION;
3635:        VAR LCP,LCP1,LCP2: CTP; LSP: STP; LSIZE: ADDRRANGE;
3636:      BEGIN
3637:        IF SY <> IDENT THEN
3638:          BEGIN ERROR(2); SKIP(FSYS + [IDENT]) END;
3639:        WHILE SY = IDENT DO
3640:          BEGIN NEW(LCP,TYPES);
3641:            WITH LCP^ DO
3642:              BEGIN NAME := ID; IDTYPE := NIL; KLASS := TYPES END;
3643:            INSYMBOL;
3644:            IF (SY = RELOP) AND (OP = EQOP) THEN INSYMBOL ELSE ERROR(16);
3645:            TYP(FSYS + [SEMICOLON],LSP,LSIZE);
3646:            ENTERID(LCP);
3647:            LCP^.IDTYPE := LSP;
3648:            LCP1 := FWPTR;
3649:            WHILE LCP1 <> NIL DO
3650:              BEGIN
3651:                IF LCP1^.NAME = LCP^.NAME THEN
3652:                  BEGIN
3653:                    LCP1^.IDTYPE^.ELTYPE := LCP^.IDTYPE;
3654:                    IF LCP1 <> FWPTR THEN
3655:                      LCP2^.NEXT := LCP1^.NEXT
3656:                    ELSE FWPTR := LCP1^.NEXT;
3657:                  END;
3658:                LCP2 := LCP1; LCP1 := LCP1^.NEXT
3659:              END;
3660:            IF SY = SEMICOLON THEN
3661:              BEGIN INSYMBOL;
3662:                IF NOT (SY IN FSYS + [IDENT]) THEN
3663:                  BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
3664:              END
3665:            ELSE ERROR(14)
3666:          END;
3667:        IF FWPTR <> NIL THEN
3668:          BEGIN ERROR(117); FWPTR := NIL END
3669:      END (*TYPEDECLARATION*) ;
3670:
3671:      PROCEDURE VARDECLARATION;
3672:        VAR LCP,NXT,IDLIST: CTP; LSP: STP; LSIZE: ADDRRANGE;
3673:      BEGIN NXT := NIL;
3674:        REPEAT
3675:          REPEAT
3676:            IF SY = IDENT THEN
3677:              BEGIN NEW(LCP,VARS);
3678:                WITH LCP^ DO
3679:                 BEGIN NAME := ID; NEXT := NXT; KLASS := VARS;
3680:                    IDTYPE := NIL; VKIND := ACTUAL; VLEV := LEVEL
3681:                  END;
3682:                ENTERID(LCP);
3683:                NXT := LCP;
3684:                INSYMBOL;
3685:              END
3686:            ELSE ERROR(2);
3687:            IF NOT (SY IN FSYS + [COMMA,COLON] + TYPEDELS) THEN
3688:              BEGIN ERROR(6); SKIP(FSYS+[COMMA,COLON,SEMICOLON]+TYPEDELS) END;
3689:            TEST := SY <> COMMA;
3690:            IF NOT TEST THEN INSYMBOL
3691:          UNTIL TEST;
3692:          IF SY = COLON THEN INSYMBOL ELSE ERROR(5);
3693:          IDLIST := NXT;
3694:          TYP(FSYS + [SEMICOLON] + TYPEDELS,LSP,LSIZE);
3695:          WHILE NXT <> NIL DO
3696:            WITH  NXT^ DO
```

```
3697:              BEGIN IDTYPE := LSP; VADDR := LC;
3698:                LC := LC + LSIZE; NXT := NEXT;
3699:                IF NEXT = NIL THEN
3700:                  IF LSP <> NIL THEN
3701:                    IF LSP^.FORM = FILES THEN
3702:                        BEGIN (*PUT IDLIST INTO LOCAL FILE LIST*)
3703:                          NEXT := DISPLAY[TOP].FFILE;
3704:                          DISPLAY[TOP].FFILE := IDLIST
3705:                        END
3706:              END;
3707:            IF SY = SEMICOLON THEN
3708:              BEGIN INSYMBOL;
3709:                IF NOT (SY IN FSYS + [IDENT]) THEN
3710:                  BEGIN ERROR(6); SKIP(FSYS + [IDENT]) END
3711:              END
3712:            ELSE ERROR(14)
3713:          UNTIL (SY <> IDENT) AND NOT (SY IN TYPEDELS);
3714:        IF FWPTR <> NIL THEN
3715:            BEGIN ERROR(117); FWPTR := NIL END
3716:        END (*VARDECLARATION*) ;
3717:
3718:        PROCEDURE PROCDECLARATION(FSY: SYMBOL);
3719:          VAR OLDLEV: 0..MAXLEVEL; LSY: SYMBOL; LCP,LCP1: CTP; LSP: STP;
3720:              FORW: BOOLEAN; OLDTOP: DISPRANGE; OLDPROC: PROCRANGE;
3721:              LLC,LCM: ADDRRANGE;  MARKP: ^INTEGER;
3722:
3723:          PROCEDURE PARAMETERLIST(FSY: SETOFSYS; VAR FPAR: CTP; FCP: CTP);
3724:            VAR LCP,LCP1,LCP2,LCP3: CTP; LSP: STP; LKIND: IDKIND;
3725:              LLC,LEN : ADDRRANGE; COUNT : INTEGER;
3726:          BEGIN LCP1 := NIL; LLC := LC;
3727:            IF NOT (SY IN FSY + [LPARENT]) THEN
3728:              BEGIN ERROR(7); SKIP(FSYS + FSY + [LPARENT]) END;
3729:            IF SY = LPARENT THEN
3730:              BEGIN IF FORW THEN ERROR(119);
3731:                INSYMBOL;
3732:                IF NOT (SY IN [IDENT,VARSY]) THEN
3733:                  BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END;
3734:                WHILE SY IN [IDENT,VARSY] DO
3735:                  BEGIN
3736:                    IF SY = VARSY THEN
3737:                      BEGIN LKIND := FORMAL; INSYMBOL END
3738:                    ELSE LKIND := ACTUAL;
3739:                    LCP2 := NIL;
3740:                    COUNT := 0;
3741:                    REPEAT
3742:                      IF SY = IDENT THEN
3743:                        BEGIN NEW(LCP,VARS);
3744:                          WITH LCP^ DO
3745:                            BEGIN NAME := ID; IDTYPE := NIL;
3746:                              VKIND := LKIND; NEXT := LCP2;
3747:                              KLASS := VARS; VLEV := LEVEL
3748:                            END;
3749:                          ENTERID(LCP);
3750:                          LCP2 := LCP; COUNT := COUNT + 1;
3751:                          INSYMBOL
3752:                        END;
3753:                      IF NOT (SY IN FSYS + [COMMA,COLON]) THEN
3754:                        BEGIN ERROR(7);
3755:                          SKIP(FSYS + [COMMA,SEMICOLON,RPARENT,COLON])
3756:                        END;
3757:                      TEST := SY <> COMMA;
3758:                      IF NOT TEST THEN INSYMBOL
3759:                    UNTIL TEST;
3760:                    IF SY = COLON THEN
3761:                      BEGIN INSYMBOL;
3762:                        IF SY = IDENT THEN
```

```
3763:                       BEGIN
3764:                         SEARCHID([TYPES],LCP);
3765:                         LSP := LCP^.IDTYPE;
3766:                         LCP3 := LCP2;
3767:                         LEN := PTRSIZE;
3768:                         IF LSP <> NIL THEN
3769:                           IF LKIND = ACTUAL THEN
3770:                             IF LSP^.FORM = FILES THEN ERROR(121)
3771:                             ELSE
3772:                               IF LSP^.FORM <= POWER THEN LEN := LSP^.SIZE;
3773:                         LC := LC + COUNT * LEN;
3774:                         WHILE LCP2 <> NIL DO
3775:                           BEGIN LCP := LCP2;
3776:                             WITH LCP2^ DO
3777:                               BEGIN IDTYPE := LSP;
3778:                                 LCP2 := NEXT
3779:                               END
3780:                           END;
3781:                         LCP^.NEXT := LCP1; LCP1 := LCP3;
3782:                         INSYMBOL
3783:                       END
3784:                     ELSE ERROR(2);
3785:                     IF NOT (SY IN FSYS + [SEMICOLON,RPARENT]) THEN
3786:                       BEGIN ERROR(7); SKIP(FSYS + [SEMICOLON,RPARENT]) END;
3787:                   END
3788:                 ELSE ERROR(5);
3789:                 IF SY = SEMICOLON THEN
3790:                   BEGIN INSYMBOL;
3791:                     IF NOT (SY IN FSYS + [IDENT,VARSY]) THEN
3792:                       BEGIN ERROR(7); SKIP(FSYS + [IDENT,RPARENT]) END
3793:                   END
3794:               END (*WHILE*) ;
3795:             IF SY = RPARENT THEN
3796:               BEGIN INSYMBOL;
3797:                 IF NOT (SY IN FSY + FSYS) THEN
3798:                   BEGIN ERROR(6); SKIP(FSY + FSYS) END
3799:               END
3800:             ELSE ERROR(4);
3801:             FCP^.LOCALLC := LC; LCP3 := NIL;
3802:             WHILE LCP1 <> NIL DO
3803:               WITH LCP1^ DO
3804:                 BEGIN LCP2 := NEXT; NEXT := LCP3;
3805:                   IF (KLASS = VARS) AND (IDTYPE <> NIL) THEN
3806:                     IF (IDTYPE^.FORM <= POWER) OR (VKIND = FORMAL) THEN
3807:                       BEGIN VADDR := LLC;
3808:                         IF VKIND = FORMAL THEN LLC := LLC + PTRSIZE
3809:                         ELSE LLC := LLC + IDTYPE^.SIZE
3810:                       END
3811:                     ELSE
3812:                       BEGIN VADDR := LC;
3813:                         LC := LC + IDTYPE^.SIZE;
3814:                         LLC := LLC + PTRSIZE
3815:                       END;
3816:                   LCP3 := LCP1; LCP1 := LCP2
3817:                 END;
3818:             FPAR := LCP3
3819:           END
3820:             ELSE FPAR := NIL
3821:     END (*PARAMETERLIST*) ;
3822:
3823:     BEGIN (*PROCDECLARATION*)
3824:       LLC := LC; LC := LCAFTERMARKSTACK;
3825:       IF FSY = FUNCSY THEN LC := LC + REALSIZE;
3826:       LINEINFO := LC; DP := TRUE;
3827:       IF SY = IDENT THEN
3828:         BEGIN SEARCHSECTION(DISPLAY[TOP].FNAME,LCP);
```

```
3829:            IF LCP <> NIL THEN
3830:             BEGIN
3831:              IF LCP^.KLASS = PROC THEN
3832:                FORW := LCP^.FORWDECL AND (FSY = PROCSY)
3833:                        AND (LCP^.PFKIND = ACTUAL)
3834:              ELSE
3835:                IF LCP^.KLASS = FUNC THEN
3836:                  FORW := LCP^.FORWDECL AND (FSY = FUNCSY)
3837:                          AND (LCP^.PFKIND = ACTUAL)
3838:                ELSE FORW := FALSE;
3839:              IF NOT FORW THEN ERROR(160)
3840:             END
3841:            ELSE FORW := FALSE;
3842:            IF NOT FORW THEN
3843:              BEGIN
3844:                IF FSY = PROCSY THEN NEW(LCP,PROC,DECLARED,ACTUAL)
3845:                ELSE NEW(LCP,FUNC,DECLARED,ACTUAL);
3846:                WITH LCP^ DO
3847:                  BEGIN NAME := ID; IDTYPE := NIL; LOCALLC := LC;
3848:                    PFDECKIND := DECLARED; PFKIND := ACTUAL;
3849:                    INSCOPE := FALSE; PFLEV := LEVEL;
3850:                    PFNAME := NEXTPROC; PFSEG := SEG;
3851:                    IF NEXTPROC = MAXPROCNUM THEN ERROR(251)
3852:                    ELSE NEXTPROC := NEXTPROC + 1;
3853:                    IF FSY = PROCSY THEN KLASS := PROC
3854:                    ELSE KLASS := FUNC
3855:                  END;
3856:                ENTERID(LCP)
3857:              END
3858:            ELSE
3859:              BEGIN LCP1 := LCP^.NEXT;
3860:                WHILE LCP1 <> NIL DO
3861:                  BEGIN
3862:                    WITH LCP1^ DO
3863:                      IF KLASS = VARS THEN
3864:                        IF IDTYPE <> NIL THEN
3865:                          BEGIN
3866:                            IF VKIND = FORMAL THEN LCM := VADDR + PTRSIZE
3867:                            ELSE LCM := VADDR + IDTYPE^.SIZE;
3868:                            IF LCM > LC THEN LC := LCM
3869:                          END;
3870:                    LCP1 := LCP1^.NEXT
3871:                  END
3872:                END;
3873:              INSYMBOL
3874:            END
3875:          ELSE
3876:            BEGIN ERROR(2); LCP := UPRCPTR END;
3877:          OLDLEV := LEVEL; OLDTOP := TOP; OLDPROC := CURPROC;
3878:          CURPROC := LCP^.PFNAME;
3879:          IF LEVEL < MAXLEVEL THEN LEVEL := LEVEL + 1 ELSE ERROR(251);
3880:          IF TOP < DISPLIMIT THEN
3881:            BEGIN TOP := TOP + 1;
3882:              WITH DISPLAY[TOP] DO
3883:                BEGIN
3884:                  IF FORW THEN FNAME := LCP^.NEXT
3885:                  ELSE FNAME := NIL;
3886:                  FLABEL := NIL; FFILE := NIL; OCCUR := BLCK
3887:                END
3888:            END
3889:          ELSE ERROR(250);
3890:          IF FSY = PROCSY THEN
3891:            BEGIN PARAMETERLIST([SEMICOLON],LCP1,LCP);
3892:              IF NOT FORW THEN LCP^.NEXT := LCP1
3893:            END
3894:          ELSE
```

```
3895:          BEGIN PARAMETERLIST([SEMICOLON,COLON],LCP1,LCP);
3896:            IF NOT FORW THEN LCP^.NEXT := LCP1;
3897:            IF SY = COLON THEN
3898:              BEGIN INSYMBOL;
3899:                IF SY = IDENT THEN
3900:                  BEGIN IF FORW THEN ERROR(122);
3901:                    SEARCHID([TYPES],LCP1);
3902:                    LSP := LCP1^.IDTYPE;
3903:                    LCP^.IDTYPE := LSP;
3904:                    IF LSP <> NIL THEN
3905:                      IF NOT (LSP^.FORM IN [SCALAR,SUBRANGE,POINTER]) THEN
3906:                        BEGIN ERROR(120); LCP^.IDTYPE := NIL END;
3907:                    INSYMBOL
3908:                  END
3909:                ELSE BEGIN ERROR(2); SKIP(FSYS + [SEMICOLON]) END
3910:              END
3911:            ELSE
3912:              IF NOT FORW THEN ERROR(123)
3913:          END;
3914:        IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
3915:        IF SY = FORWARDSY THEN
3916:          BEGIN
3917:            IF FORW THEN ERROR(161)
3918:            ELSE LCP^.FORWDECL := TRUE;
3919:            INSYMBOL;
3920:            IF SY = SEMICOLON THEN INSYMBOL ELSE ERROR(14);
3921:            IF NOT (SY IN FSYS) THEN
3922:              BEGIN ERROR(6); SKIP(FSYS) END
3923:          END
3924:        ELSE
3925:          BEGIN MARK(MARKP);
3926:            WITH LCP^ DO
3927:              BEGIN FORWDECL := FALSE; INSCOPE := TRUE END;
3928:            REPEAT BLOCK(FSYS,SEMICOLON,LCP);
3929:              RELEASE(MARKP);
3930:              IF SY = SEMICOLON THEN
3931:                BEGIN INSYMBOL;
3932:                  IF NOT (SY IN [BEGINSY,PROCSY,FUNCSY,PROGSY]) THEN
3933:                    BEGIN ERROR(6); SKIP(FSYS) END
3934:                END
3935:              ELSE ERROR(14)
3936:            UNTIL SY IN [BEGINSY,PROCSY,FUNCSY,PROGSY];
3937:            LCP^.INSCOPE := FALSE
3938:          END;
3939:        LEVEL := OLDLEV; TOP := OLDTOP; LC := LLC; CURPROC := OLDPROC
3940:      END (*PROCDECLARATION*) ;
3941:
3942:      PROCEDURE SEGDECLARATION;
3943:        VAR LSY: SYMBOL; OLDPROC: PROCRANGE; OLDSEG: SEGRANGE;
3944:      BEGIN
3945:        IF CODEINSEG THEN
3946:          BEGIN ERROR(399); SEGINX := 0; CURBYTE := 0 END;
3947:        OLDSEG := SEG; SEG := NEXTSEG; OLDPROC := NEXTPROC;
3948:        IF NEXTSEG > MAXSEG THEN ERROR(250)
3949:        ELSE NEXTSEG := NEXTSEG + 1;
3950:        NEXTPROC := 1; LSY := SY;
3951:        IF SY IN [PROCSY,FUNCSY] THEN INSYMBOL
3952:        ELSE
3953:          BEGIN ERROR(399); LSY := PROCSY END;
3954:        IF SY = IDENT THEN SEGTABLE[SEG].SEGNAME := ID;
3955:        PROCDECLARATION(LSY);
3956:        IF CODEINSEG THEN FINISHSEG;
3957:        NEXTPROC := OLDPROC; SEG := OLDSEG
3958:      END (*SEGDECLARATION*) ;
3959:
3960:      PROCEDURE BODY(FSYS: SETOFSYS);
```

```
3961:       VAR LLC1,EXITIC: ADDRRANGE; LCP,LLCP: CTP; LOP: OPRANGE;
3962:           LLP: LABELP; LMIN,LMAX: INTEGER; JTINX: JTABRANGE;
3963:
3964:     BEGIN NEXTJTAB := 1; WRITELN(OUTPUT);
3965:       IF FPROCP = NIL THEN WRITELN(OUTPUT,'SYSTEM')
3966:       ELSE
3967:         BEGIN WRITELN(OUTPUT,FPROCP^.NAME);
3968:           LLC1 := FPROCP^.LOCALLC; LCP := FPROCP^.NEXT;
3969:           WHILE LCP <> NIL DO
3970:             WITH LCP^ DO
3971:               BEGIN
3972:                 IF KLASS = VARS THEN
3973:                   IF IDTYPE <> NIL THEN
3974:                     IF (VKIND = ACTUAL) AND (IDTYPE^.FORM > POWER) THEN
3975:                       BEGIN LLC1 := LLC1 - PTRSIZE;
3976:                         GEN2(50(*LDA*),0,VADDR);
3977:                         GEN2(54(*LOD*),0,LLC1);
3978:                         IF PAOFCHAR(IDTYPE) THEN
3979:                           WITH IDTYPE^ DO
3980:                             IF AISSTRNG THEN GEN1(42(*SAS*),MAXLENG)
3981:                             ELSE
3982:                               IF INXTYPE <> NIL THEN
3983:                                 BEGIN GETBOUNDS(INXTYPE,LMIN,LMAX);
3984:                                   GEN1(41(*MVB*),LMAX - LMIN + 1)
3985:                                 END
3986:                             ELSE
3987:                         ELSE GEN1(40(*MOV*),IDTYPE^.SIZE)
3988:                       END
3989:                     ELSE
3990:                       IF VKIND = FORMAL THEN LLC1 := LLC1 - PTRSIZE
3991:                       ELSE LLC1 := LLC1 - IDTYPE^.SIZE;
3992:                 LCP := NEXT
3993:               END;
3994:         END;
3995:       WRITE(OUTPUT,'<',SCREENDOTS:4,'>');
3996:       STARTDOTS := SCREENDOTS;
3997:       LCMAX := LC;
3998:       LLP := DISPLAY[TOP].FLABEL;
3999:       WHILE LLP <> NIL DO
4000:         BEGIN GENLABEL(LLP^.CODELBP);
4001:           LLP := LLP^.NEXTLAB
4002:         END;
4003:       LCP := DISPLAY[TOP].FFILE;
4004:       WHILE LCP <> NIL DO
4005:         WITH LCP^,IDTYPE^ DO
4006:           BEGIN
4007:             GEN2(50(*LDA*),0,VADDR);
4008:             GEN2(50(*LDA*),0,VADDR+FILESIZE);
4009:             IF FILTYPE = NIL THEN GENLDC(-1)
4010:             ELSE
4011:               IF FILTYPE = CHARPTR THEN GENLDC(-2)
4012:               ELSE GENLDC(FILTYPE^.SIZE);
4013:             GEN2(77(*CXP*),0(*SYS*),3(*FINIT*));
4014:             LCP := NEXT
4015:           END;
4016:       REPEAT
4017:         REPEAT STATEMENT(FSYS + [SEMICOLON,ENDSY])
4018:         UNTIL NOT (SY IN STATBEGSYS);
4019:         TEST := SY <> SEMICOLON;
4020:         IF NOT TEST THEN INSYMBOL
4021:       UNTIL TEST;
4022:       IF SY = ENDSY THEN INSYMBOL ELSE ERROR(13);
4023:       EXITIC := IC;
4024:       LCP := DISPLAY[TOP].FFILE;
4025:       WHILE LCP <> NIL DO
4026:         WITH LCP^ DO
```

```
4027:            BEGIN
4028:              GEN2(50(*LDA*),0,VADDR);
4029:              GENLDC(0); GEN2(77(*CXP*),0(*SYS*),6(*FCLOSE*));
4030:              LCP := NEXT
4031:            END;
4032:        IF FPROCP = NIL THEN GEN0(86(*XIT*))
4033:        ELSE
4034:          BEGIN
4035:            IF FPROCP^.PFLEV = 0 THEN LOP := 65(*RBP*)
4036:            ELSE LOP := 45(*RNP*);
4037:            IF FPROCP^.IDTYPE = NIL THEN GEN1(LOP,0)
4038:            ELSE GEN1(LOP,FPROCP^.IDTYPE^.SIZE)
4039:          END;
4040:        LLP := DISPLAY[TOP].FLABEL;  (* CHECK UNDEFINED LABELS *)
4041:        WHILE LLP <> NIL DO
4042:          WITH LLP^,CODELBP^ DO
4043:            BEGIN
4044:              IF NOT DEFINED THEN
4045:                IF REFLIST <> MAXADDR THEN ERROR(168);
4046:              LLP := NEXTLAB
4047:            END;
4048:        JTINX := NEXTJTAB - 1;
4049:        IF ODD(IC) THEN IC := IC + 1;
4050:        WHILE JTINX > 0 DO
4051:          BEGIN GENWORD(IC-JTAB[JTINX]); JTINX := JTINX-1 END;
4052:        IF FPROCP = NIL THEN
4053:          BEGIN GENWORD((LCMAX-LCAFTERMARKSTACK)*2); GENWORD(0) END
4054:        ELSE
4055:          WITH FPROCP^ DO
4056:            BEGIN GENWORD((LCMAX-LOCALLC)*2);
4057:              GENWORD((LOCALLC-LCAFTERMARKSTACK)*2)
4058:            END;
4059:        GENWORD(IC-EXITIC); GENWORD(IC);
4060:        GENBYTE(CURPROC); GENBYTE(LEVEL-1);
4061:        IF NOT CODEINSEG THEN
4062:          BEGIN CODEINSEG := TRUE;
4063:            SEGTABLE[SEG].DISKADDR := CURBLK
4064:          END;
4065:        WRITECODE(FALSE);
4066:        SEGINX := SEGINX + IC;
4067:        PROCTABLE[CURPROC] := SEGINX - 2
4068:      END (*BODY*) ;
4069:
4070:    PROCEDURE FINDFORW(FCP: CTP);
4071:    BEGIN
4072:      IF FCP <> NIL THEN
4073:        WITH FCP^ DO
4074:          BEGIN
4075:            IF KLASS IN [PROC,FUNC] THEN
4076:              IF PFDECKIND = DECLARED THEN
4077:                IF PFKIND = ACTUAL THEN
4078:                  IF FORWDECL THEN
4079:                    BEGIN
4080:                      USERINFO.ERRNUM := 117; WRITELN(OUTPUT);
4081:                      WRITE(OUTPUT,NAME,' Undefined')
4082:                    END;
4083:            FINDFORW(RLINK); FINDFORW(LLINK)
4084:          END
4085:    END (*FINDFORW*) ;
4086:
4087:    BEGIN (*BLOCK*)
4088:      REPEAT
4089:        IF SY = LABELSY THEN
4090:          BEGIN INSYMBOL; LABELDECLARATION END;
4091:        IF SY = CONSTSY THEN
4092:          BEGIN INSYMBOL; CONSTDECLARATION END;
```

```
4093:        IF SY = TYPESY THEN
4094:          BEGIN INSYMBOL; TYPEDECLARATION END;
4095:        IF SY = VARSY THEN
4096:          BEGIN INSYMBOL; VARDECLARATION END;
4097:        WHILE SY IN [PROCSY,FUNCSY,PROGSY] DO
4098:          BEGIN LSY := SY; INSYMBOL;
4099:            IF LSY = PROGSY THEN SEGDECLARATION
4100:            ELSE PROCDECLARATION(LSY,FALSE)
4101:          END;
4102:        IF SY <> BEGINSY THEN
4103:          IF NOT (INCLUDING AND
4104:              (SY IN [LABELSY,CONSTSY,TYPESY,VARSY,PROCSY,FUNCSY,PROGSY])) THEN
4105:            BEGIN ERROR(18); SKIP(FSYS) END
4106:      UNTIL SY IN STATBEGSYS;
4107:      DP := FALSE; IC := 0; LINEINFO := 0;
4108:      IF SY = BEGINSY THEN INSYMBOL ELSE ERROR(17);
4109:      IF NOT SYSCOMP THEN FINDFORW(DISPLAY[TOP].FNAME);
4110:      REPEAT BODY(FSYS + [CASESY]);
4111:        IF SY <> FSY THEN
4112:          BEGIN ERROR(6); SKIP(FSYS + [FSY]) END
4113:      UNTIL (SY = FSY) OR (SY IN BLOCKBEGSYS);
4114:    END (*BLOCK*) ;
4115:
4116:  BEGIN (*COMPILER*)
4117:    COMPINIT; TIME(LGTH,LOWTIME);
4118:    BLOCK(BLOCKBEGSYS+STATBEGSYS-[CASESY],PERIOD,OUTERBLOCK);
4119:    IF SY <> PERIOD THEN ERROR(21);
4120:    IF LIST THEN
4121:      BEGIN SCREENDOTS := SCREENDOTS+1;
4122:        SYMBUFP^[SYMCURSOR] := CHR(EOL);
4123:        SYMCURSOR := SYMCURSOR+1; PRINTLINE
4124:      END;
4125:    FINISHSEG;
4126:    TIME(LGTH,STARTDOTS); LOWTIME := STARTDOTS-LOWTIME;
4127:    UNITWRITE(3,IC,7); WRITELN(OUTPUT);
4128:    WRITE(OUTPUT,SCREENDOTS,' lines');
4129:    IF LOWTIME > 0 THEN
4130:      WRITE(OUTPUT,', ',(LOWTIME+30) DIV 60,' secs, ',
4131:          ROUND((3600/LOWTIME)*SCREENDOTS),' lines/min');
4132:    IC := 0;
4133:    FOR SEG := 0 TO MAXSEG DO
4134:      WITH SEGTABLE[SEG] DO
4135:        BEGIN GENWORD(DISKADDR); GENWORD(CODELENG) END;
4136:    FOR SEG := 0 TO MAXSEG DO
4137:      WITH SEGTABLE[SEG] DO
4138:        FOR LGTH := 1 TO 8 DO
4139:          GENBYTE(ORD(SEGNAME[LGTH]));
4140:    CURBLK := 0; CURBYTE := 0; WRITECODE(TRUE)
4141:  END (*COMPILE*) ;
4142:
4143:  BEGIN END.
4144:

THAT'S ALL FOLKS!      LINES: 4144   CHARACTERS: 146027
```