```
Pass 1.1
 *** Compilers 16/2/77


    let Pass1[] be
    §P1
        Send := Write
 5      SpecSend := Write
        SetupStructure[]
        SetupNameVec[]
        Peep[1]
        PassNo := 1
10      LineNo := 1
        GetNo := 0
        ReportNo := 0
        GN, SN, MFN := 0, 0, 0
        Send[ENDPROG]
15      NewLevel[RTBODY]
        ExpError := false
        UpdateCh[SECTBRA]
        Block[]
        unless Ch = ENDPROG do
20          Error[6, HARD, Ch, NullProgram]
        ReturnMap[NameMap]
        Peep[3]
    §P1

25  and SetupStructure[] be
    §SLV
        if StructureSize < 20 do
            CompilerError[1101]
        StructureVec := TopVec[StructureSize]
30      StructureLPtr := 0
        StructureNPtr := StructureSize
        NILNAME := FormNameBlock[UNDEFINED, UNDEFINED, 0, UNDEFINED,
                UNDEFINED, UNDEFINED, UNDEFINED]
        UpdateNextName[NILNAME, NILNAME]
35      UpdatePreviousNameBlock[NILNAME, NILNAME]
        OUTSIDE := FormLevel[UNDEFINED, UNDEFINED, 0, NILNAME]
        UpdatePreviousLevel[OUTSIDE, OUTSIDE]
        PresentLevel := OUTSIDE
        LocalGenNo := 0
40      LastNL := OUTSIDE - StructureVec
    §SLV

    and SetupNameVec[] be
    §SNV
45      let Size = NameSize + 26
        NameVec := TopVec[Size]
        for i = 0 to Size do NameVec↓i := NILNAME
    §SNV

50
    and TopVec[n] = valof
    §TV
        let m = MaxVecSize[]
        let v = NewVec[m]
55      ReturnVec[v, m - n - 1]
        resultis v + m - n
    §TV

    and LevelChange[] be
60  §LC
        let PresNL = PresentLevel - StructureVec
```

```
        Send[NEWLEVEL \/ LastNL]
        Send[NEWLEVEL \/ PresNL]
        LastNL := PresNL
65  $LC

    and Block[] be
    §B
        unless ChType = SECTBRA do
70          Error[111, HARD, Ch, BackUp, SECTBRA]
      § let SecKet, Level = FormSecKet[Ch], PresentLevel
        and DefBlock = false
        Scan[]

75      §r
            switchon ChType into
            §s
                case GLOBAL:
                        DefBlock := true
80                      ReadDefBlock[Ch, COLON]
                        endcase
                case MANIFEST:
                case STATIC:
                case TABLE:
85                      DefBlock := true
                        ReadDefBlock[Ch, EQ]
                        endcase
                case AND:
                        Error[105, HARD, Ch, UpdateCh, LET]
90              case LET:
                        DefBlock := true
                        ReadDef[]
                        endcase
                case SEMICOLON:
95                      Scan[]
                default:
                        §r
                            Command[]
                            if ChType = SECTKET  \/  Ch = ENDPROG break
100                         if StartsDef[] do
                            §   Error[142, HARD, Ch, NullProgram]
                                if Ch = AND do UpdateCh[LET]
                                BackUp[SECTBRA]
                                BackUp[SEMICOLON]
105                         §
                            unless Ch = SEMICOLON do
                            §   Send[SEMICOLON]
                                Error[142, HARD, Ch, DeleteCommand]
                                unless Ch = SEMICOLON break
110                         §
                            Scan[]
                        $r repeat
                case SECTKET:
                case ENDPROG:
115                     test Ch = SecKet
                          ifso  Scan[]
                          ifnot InsertSecKet[SecKet]
                        until PresentLevel = Level do UpOneLevel[]
                        if DefBlock do LevelChange[]
120                     return
            $s
        $r repeat
    $B

125 and ReadDefBlock[DefType, Delim] be
    §RDB
```

```
        let SecKet = SECTKET
        and SaveI = Ignoring
        NewLevel[LOCAL]
130     Read[]
        test ChType = SECTBRA
          ifso  §   SecKet := FormSecKet[Ch]
                    Scan[]
                §
135       ifnot Send[SECTBRA]

        §r
            let N = Name[]
            Ignoreif[(DefType = MANIFEST ∨ DefType = GLOBAL) ∧
140                  EntryinMap[NameMap, ValPart[N]]]
            test N = NOTNAME
              ifso  Error[201, HARD, Ch, DeleteCommand]
              ifnot switchon Ch into
                    §s
145                     case COLON:
                        case EQ:
                                unless Ch = Delim do
                                §   Error[105, HARD, Ch, DeleteCommand]
                                    endcase
150                             §
                                UpdateCh[DUMMY]
                                AddDef[N, DefType, DefType]
                                ScanExpression[] repeatwhile Ch = COMMA ∧
                                        DefType = TABLE
155                             endcase

                        case SBRA:
                                unless DefType = MANIFEST do
                                §   Error[105, HARD, Ch, DeleteCommand]
160                                 endcase
                                §
                                ProcDef[N, MANFN, MANFNDF, IS]
                                endcase

165                     default:Error[105, HARD, Ch, DeleteCommand]
                    §s

            if SecKet = SECTKET do BackUp[SECTKET]
            Ignoreif[SaveI]
170         if ChType = SECTKET  ∨  Ch = ENDPROG break
            unless Ch = SEMICOLON do
            §   test StartsDef[]
                  ifso  Error[142, HARD, Ch, BackUp, SecKet]
                  ifnot §   Send[SEMICOLON]
175                         Error[142, HARD, Ch, DeleteCommand]
                        §
                unless Ch = SEMICOLON break
            §
            Scan[]
180     $r repeat

        test Ch = SecKet
          ifso  Scan[]
          ifnot InsertSecKet[SecKet]
185 §RDB


    and InsertSecKet[SecKet] be
    §ISK
190     unless SecKet = SECTKET do
          test Ch = ENDPROG
```

```
            ifso  Error[5, HARD, Ch, NullProgram]
            ifnot unless SecKet = NullSecKet[] do
                    Error[107, SOFT, SecKet, NullProgram]
195     Send[SecKet]
    §ISK


    and ReadDef[] be
200 §RD
        NewLevel[LOCAL]

        §r1
            let n = 1
205         §r2
                let N = ScanName[]
                if N = NOTNAME do
                §   Error[201, HARD, Ch, DeleteCommand]
                    break
210             §
                switchon Ch into
                §s
                    case COMMA:
                            AddDef[N, SIMPLE, VALDF]
215                         n := n+1
                            endcase

                    case EQ:
                            Read[]
220                         if Ch = VEC do
                            §   AddDef[N, SIMPLE, VECDF]
                                unless n = 1 do
                                §   Error[303, HARD, Ch, DeleteCommand]
                                    break
225                             §
                                Read[]
                                Expression[]
                                break
                            §
230                         AddDef[N, SIMPLE, VALDF]
                            Send[DEF]
                            §r3
                                Expression[]
                                n := n-1
235                             unless Ch = COMMA break
                                Scan[]
                            §r3 repeat
                            unless n = 0 do
                                Error[301, HARD, Ch, DeleteCommand]
240                         break

                    case SBRA:
                            ProcDef[N, STATIC, FNDF, BE]
                            break
245
                    default:
                            Error[105, HARD, Ch, DeleteCommand]
                            break
                §s
250         §r2 repeat

            if StartsDef[] loop
            if Ch = SEMICOLON ∨ ChType = SECTKET ∨ Ch = ENDPROG break
            Error[105, HARD, Ch, DeleteCommand]
255
        §r1 repeatwhile Ch = AND
```

```
    $RD

260 and ProcDef[N, AddType, DefType, Delim] be
    §PD
        AddDef[N, AddType, DefType]
        NewLevel[FNBODY]
        ParameterList[]
265     unless Ch = SKET do
        §   Error[116, HARD, Ch, DelComRestLevel]
            return
        $
        Read[]
270     switchon Ch into
        §s
            case EQ:
                    Send[FNBODY]
                    Read[]
275                 Expression[]
                    unless ExpError do Send[ENDBODY]
                    endcase

            case BE:
280         case IS:
                    unless Ch = Delim do
                        Error[105, HARD, Ch, NullProgram]
                    Send[RTBODY]
                    Read[]
285                 UpdateLevelType[PresentLevel, RTBODY]
                    Command[]
                    Send[ENDBODY]
                    endcase

290         default:
                    Error[105, HARD, Ch, DeleteCommand]
        $s
        RestoreLevel[]
    $PD
295
    and Name[] = valof
    §N
        unless IsName[Ch] resultis NOTNAME
      § let N = Ch
300     let P = NameBlock[N]
        unless Ignoring do
            if Generation[P] = Generation[PresentLevel] do
                Error[202, HARD, Ch, NullProgram]
        Read[]
305     resultis N
    $N

    and NewLevel[LType] be
        unless Ignoring do
310     §NL
            let L = FormLevel[PresentLevel, LType,
                    Generation[PresentLevel] + 2, NILNAME]
            PresentLevel := L
            LevelChange[]
315     $NL

    and RestoreLevel[] be
        unless Ignoring do
        §RL
320         UpOneLevel[]
            LevelChange[]
```

```
        $RL

    and ParameterList[] be
325 §PL
        Read[]
        if Ch = SKET return
        §r
            let N = Name[]
330         if N = NOTNAME do
            §   Error[201, HARD, Ch, BackUp, ERROR]
                break
            §
            AddName[N, SIMPLE]
335         OutputDef[N, PARAM]
            unless Ch = COMMA break
            Scan[]
        $r repeat
    $PL
340
    and StartsDef[] = valof
    §SD
        switchon ChType into
        §s
345         case LET:
            case AND:
            case GLOBAL:
            case MANIFEST:
            case TABLE:
350         case STATIC:
                    resultis true

            default:resultis false
        $s
355 $SD

****
```