1. THE ALGOL 60 PROGRAM

```
begin
comment     ALGOL 60 - version of the ALGOL 60 - translator
            for the EL - X8, F.E.J. Kruseman Aretz;

comment     basic symbols;
integer plus, minus, mul, div, idi, ttp, equ, uqu, les, mst, mor, lst,
        non, qvl, imp, or, and, goto, for, step, until, while, do,
        comma, period, ten, colon, semicolon, colonequal, space sbl,
        if, then, else, comment, open, close, sub, bus, quote, unquote,
        begin, end, own, rea, integ, boole, stri, array, proced, switch,
        label, value, true, false, new line, underlining, bar;

comment     other global integers;
integer case, lower case, stock, stock1, last symbol, line counter,
        last identifier, last identifier1,
        quote counter, run number, shift,
        type, chara, character, value character, arr decla macro,
        value of constant, decimal exponent, decimal count,
        word count, nlp, last nlp, n, integer label,
        block cell pointer, next block cell pointer,
        dimension, forcount, instruct counter, dp0,
        function letter, function digit, c variant,
        nl base, prog base, text base, text pointer,
        end of text, end of memory, start, end of list,
        d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14,
        d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25,
        re, in, bo, st, ar, nondes, des, un, arbo, intlab;

comment     macro identifiers;
integer STACK, NEG, ADD, SUB, MUL, DIV, IDI, TTP,
        EQU, UQU, LES, MST, MOR, LST, STAB, NON, QVL, IMP, OR, AND,
        STAA, TSR, TSI, TSB, TSST, TFSU, TSL, TFSL, TCST,
        STSR, STSI, SSTSI, STSB, STSST, STFSU,
        ENTRIS, TFD, SAS, DECS, FAD, TASR, TASI, TASB, TASST, TASU,
        EXITIS, FADCV, TRSCV, TISCV, TSCVU, EXIT, TEST1, TEST2,
        CRV, CIV, CBV, CSTV, CLV, CEN, CLPN, TAV, TIAV,
        RAD, IAD, BAD, STAD, ORAD, OIAD, OBAD, OSTAD,
        LOS, EXITP, EXITPC, REJST, JUA, EMPTY,
        ABS, SIGN, ENTIER, SQRT, EXP, LN, END;

comment     macro2 identifiers;
integer TRV, TIV, TRC, TIC, TSIC, TBV, TBC, TSTV, TLV, TAK, TSWE,
        STR, STI, SSTI, STB, STST, DOS, DOS2, DOS3,
        JU, JU1, LJU, LJU1, COJU, YCOJU, SUBJ, ISUBJ, DECB, DO,
        TBL, ENTRB, DPTR, INCRB, TDL, ENTRPB, NIL, LAST,
        LAD, TDA, TNA, TAA, SWP, EXITB, EXITC, EXITSV,
        CODE, SLNC, RLNC, LNC;

comment     global Booleans;
Boolean letter last symbol, digit last symbol, arr declarator last symbol,
        type declarator last symbol, in array declaration, in formal list,
        text in memory, own type, int labels, real number, small,
        erroneous, derroneous, wanted;
```

```
comment         global arrays;
integer array internal representation[0 : 127], word delimiter[0 : 23],
              macro list[0 : 511], tabel[5 : 59],
              instruct list[0 : 203], mask[0 : 9];

comment         start of initialization;
        plus:= read;    minus:= read;   mul:= read;     div:= read;
        idi:= read;     ttp:= read;     equ:= read;     uqu:= read;
        les:= read;     mst:= read;     mor:= read;     lst:= read;
        non:= read;     qvl:= read;     imp:= read;     or:= read;
        and:= read;     goto:= read;    for:= read;     step:= read;
        until:= read;   while:= read;   do:= read;      comma:= read;
        period:= read;  ten:= read;     colon:= read;   semicolon:= read;
        colonequal:= read; space sbl:= read; if:= read; then:= read;
        else:= read;    comment:= read; open:= read;    close:= read;
        sub:= read;     bus:= read;     quote:= read;   unquote:= read;
        begin:= read;   end:= read;     own:= read;     rea:= read;
        integ:= read;   boole:= read;   stri:= read;    array:= read;
        proced:= read;  switch:= read;  label:= read;   value:= read;
        true:= read;    false:= read;   new line:= read;
        underlining:= read; bar:= read; lower case:= read;

        STACK:= read;   NEG:= read;     ADD:= read;     SUB:= read;
        MUL:= read;     DIV:= read;     IDI:= read;     TTP:= read;
        EQU:= read;     UQU:= read;     LES:= read;     MST:= read;
        MOR:= read;     LST:= read;     STAB:= read;    NON:= read;
        QVL:= read;     IMP:= read;     OR:= read;      AND:= read;
        STAA:= read;    TSR:= read;     TSI:= read;     TSB:= read;
        TSST:= read;    TFSU:= read;    TSL:= read;     TFSL:= read;
        TCST:= read;    STSR:= read;    STSI:= read;    SSTSI:= read;
        STSB:= read;    STSST:= read;   STFSU:= read;   ENTRIS:= read;
        TFD:= read;     SAS:= read;     DECS:= read;    FAD:= read;
        TASR:= read;    TASI:= read;    TASB:= read;    TASST:= read;
        TASU:= read;    EXITIS:= read;  FADCV:= read;   TRSCV:= read;
        TISCV:= read;   TSCVU:= read;   EXIT:= read;    TEST1:= read;
        TEST2:= read;   CRV:= read;     CIV:= read;     CBV:= read;
        CSTV:= read;    CLV:= read;     CEN:= read;     CLPN:= read;
        TAV:= read;     TIAV:= read;    RAD:= read;     IAD:= read;
        BAD:= read;     STAD:= read;    ORAD:= read;    OIAD:= read;
        OBAD:= read;    OSTAD:= read;   LOS:= read;     EXITP:= read;
        EXITPC:= read;  REJST:= read;   JUA:= read;     EMPTY:= read;
        ABS:= read;     SIGN:= read;    ENTIER:= read;  SQRT:= read;
        EXP:= read;     LN:= read;      END:= read;
```

```
        TRV:= read;     TIV:= read;     TRC:= read;     TIC:= read;
        TSIC:= read;    TBV:= read;     TBC:= read;     TSTV:= read;
        TLV:= read;     TAK:= read;     TSWE:= read;    STR:= read;
        STI:= read;     SSTI:= read;    STB:= read;     STST:= read;
        DOS:= read;     DOS2:= read;    DOS3:= read;    JU:= read;
        JU1:= read;     LJU:= read;     LJU1:= read;    COJU:= read;
        YCOJU:= read;   SUBJ:= read;    ISUBJ:= read;   DECB:= read;
        DO:= read;      TBL:= read;     ENTRB:= read;   DPTR:= read;
        INCRB:= read;   TDL:= read;     ENTRPB:= read;  NIL:= read;
        LAST:= read;    LAD:= read;     TDA:= read;     TNA:= read;
        TAA:= read;     SWP:= read;     EXITB:= read;   EXITC:= read;
        EXITSV:= read;  CODE:= read;    SLNC:= read;    RLNC:= read;
        LNC:= read;

        d0 :=        1; d1 :=        2; d2 :=        4; d3 :=        8;
        d4 :=       16; d5 :=       32; d6 :=       64; d7 :=      128;
        d8 :=      256; d9 :=      512; d10:=     1024; d11:=     2048;
        d12:=     4096; d13:=     8192; d14:=    16384; d15:=    32768;
        d16:=    65536; d17:=   131072; d18:=   262144; d19:=   524288;
        d20:=  1048576; d21:=  2097152; d22:=  4194304; d23:=  8388608;
        d24:= 16777216; d25:= 33554432;

        re:= 0;         in:= 1;         bo:= 2;         st:= 3;
        ar:= 4;         nondes:= 5;     des:= 6;        un:= 7;
        arbo:= 8;       intlab:= 9;

        function letter:= read; function digit:= read; c variant:= read;

        for n:= 0 step 1 until 127 do internal representation[n]:= read;
        for n:= 0 step 1 until  23 do word delimiter[n]:= read;
        for n:= 0 step 1 until 511 do macro list[n]:= read;
        for n:= 5 step 1 until  59 do tabel[n]:= read;
        for n:= 0 step 1 until 203 do instruct list[n]:= read;
        for n:= 0 step 1 until   9 do mask[n]:= d20 × read;

        end of memory:= read;
        end of list:= instruct list[174];
        text in memory:= true; erroneous:= derroneous:= false;
        wanted:= read = 0;


begin   integer array space[0 : end of memory];

procedure  ERRORMESSAGE (n); integer  n;
begin   integer  i;
        erroneous:= true;
        if n = 122 ∨ n = 123 ∨ n = 126 ∨ n = 127 ∨ n = 129
        then derroneous:= true;
        if n > run number
        then begin NLCR; PRINTTEXT (‡er‡); print (n);
                   print (line counter); print (last symbol);
                   for i:= 0 step 1 until word count do
                   print (space[nl base − last nlp − i])
             end
end  ERRORMESSAGE;
```

```
integer procedure  next symbol;
begin   integer  symbol;
next0:  symbol:= if stock1 ≥ 0 then stock1 else next basic symbol;
        stock1:= - 1;
        if (last symbol = semicolon ∨ last symbol = begin) ∧
           symbol = comment
        then begin skip0: symbol:= next basic symbol;
                          if symbol ≠ semicolon then goto skip0;
                          goto next0
             end;
        if last symbol = end
        then begin
             skip1: if symbol ≠ end ∧ symbol ≠ semicolon ∧ symbol ≠ else
                    then begin symbol:= next basic symbol; goto skip1 end
             end
        else
        if symbol = 125
        then begin stock1:= next basic symbol;
                   if stock1 > 9 ∧ stock1 < 64
                   then begin skip2: stock1:= next basic symbol;
                                     if stock1 > 9 ∧ stock1 < 64
                                     then goto skip2;
                                     if stock1 = colon
                                     then stock1:= next basic symbol
                                     else ERRORMESSAGE (100);
                                     if stock1 = open  then stock1:= - stock1
                                                       else ERRORMESSAGE (101);
                                     symbol:= comma
                              end
                   else symbol:= close
             end;
        digit last symbol := symbol < 10 ∨ symbol = period ∨
                             symbol = ten;
        letter last symbol:= symbol < 64 ∧ ¬ digit last symbol;
        next symbol:= last symbol:= symbol;
        outsymbol (run number, symbol);
        test pointers
end  next symbol;


integer procedure  next basic symbol;
begin   integer  symbol;
next0:  insymbol (run number, symbol);
        if symbol = new line
        then begin line counter:= line counter + 1;
                   if quote counter = 0
                   then begin outsymbol (run number, symbol);
                              goto next0
                        end
             end;
        next basic symbol:= symbol
end  next basic symbol;
```

```
procedure  insymbol (source, destination); integer  source, destination;
begin   integer  symbol, i;
        if (source = 200 ∨ source = 300) ∧ text in memory
        then
        begin destination:= bit string(d8 × shift, shift,
                              space[text base + text pointer]);
              if shift < 257
              then shift:= d8 × shift
              else begin shift:= 1; text pointer:= text pointer + 1 end
        end
        else
        begin symbol:= if stock > 0 then stock else next tape symbol;
              stock:= – 1;
              if symbol > bus
              then
              begin if symbol = 123 then symbol:= space sbl;
                    if quote counter > 0
                    then
                    begin if symbol = bar
                          then
                          begin next0: stock:= next tape symbol;
                                       if stock = bar then goto next0;
                                if stock = les
                                then quote counter:= quote counter + 1
                                else
                                if stock = mor
                                then
                                begin if quote counter = 1
                                      then begin symbol:= unquote;
                                                 stock:= – symbol
                                           end
                                      else quote counter:=
                                           quote counter – 1
                                end
                          end
                          else if symbol = 124
                               then symbol:= colon
                               else if symbol = 125 then symbol:= close
                    end
                    else
                    if symbol ≥ new line
                    then
                    begin if symbol = bar
                          then
                          begin next1: symbol:= next tape symbol;
                                       if symbol = bar then goto next1;
                                symbol:= if symbol = and then ttp else
                                         if symbol = equ then uqu else
                                         if symbol = les then quote else
                                         if symbol = mor then unquote
                                                         else 160
                          end
                          else
```

```
                    if symbol = underlining
                    then
                    begin symbol:= the underlined symbol;
                          if symbol > 63
                          then symbol:=
                               if symbol = 124 then idi else
                               if symbol = les then mst else
                               if symbol = mor then lst else
                               if symbol = non then imp else
                               if symbol = equ then qvl
                                               else 161
                          else
                          begin stock:= next tape symbol;
                                if stock = underlining
                                then
                                begin
                                 symbol:= the underlined symbol +
                                          d7 × symbol;
                                 for i:= 0 step 1 until 23 do
                                 begin
                                 if word delimiter[i] : d7 = symbol
                                 then
                                  begin
                                  symbol:= word delimiter[i];
                                  symbol:= symbol -
                                           symbol : d7 × d7;
                                  goto next2
                                  end
                                 end;
                                 symbol:= 162;
                          next2: stock:= next tape symbol;
                                 if stock = underlining
                                 then
                                 begin the underlined symbol;
                                       goto next2
                                 end
                                end
                                else symbol:= 161
                          end
                    end
                    else
                    if symbol = 124
                    then begin stock:= next tape symbol;
                               if stock = equ
                               then begin symbol:= colonequal;
                                          stock:= - symbol
                                    end
                               else symbol:= colon
                          end
               end
               else insymbol (runnumber, symbol)
          end;
          destination:= symbol
     end
end  insymbol;
```

```
integer procedure  the underlined symbol;
begin   integer  symbol;
        symbol:= next tape symbol;
        the underlined symbol:= if symbol = underlining
                                then the underlined symbol
                                else symbol
end  the underlined symbol;


integer procedure  next tape symbol;
begin   integer  symbol, head;
        symbol:= internal representation[REHEP];
        if symbol > 0
        then begin head:= symbol : d8;
                   next tape symbol:= abs (if case = lower case
                                           then symbol - d8 × head
                                           else head)
             end
        else begin if symbol < - 2 then case:= - symbol else
                   if symbol =   0 then ERRORMESSAGE (102) else
                   if symbol = - 1 then ERRORMESSAGE (103);
                   next tape symbol:= next tape symbol
             end
end  next tape symbol;


procedure  outsymbol (destination, source); integer destination, source;
begin   if destination = 100 ∧ text in memory
        then begin space[text base + text pointer]:=
                   space[text base + text pointer] + shift × source;
                   if shift < 257
                   then shift:= d8 × shift
                   else begin shift:= 1; text pointer:= text pointer + 1;
                              space[text base + text pointer]:= 0
                        end
             end
end  outsymbol;


Boolean procedure  arithoperator last symbol;
begin   arithoperator last symbol:= last symbol = plus  ∨
                                    last symbol = minus ∨
                                    last symbol = mul   ∨
                                    last symbol = div   ∨
                                    last symbol = idi   ∨
                                    last symbol = ttp
end  arithoperator last symbol;
```

```
Boolean procedure  relatoperator last symbol;
begin   relatoperator last symbol:= last symbol = les ∨
                                    last symbol = mst ∨
                                    last symbol = equ ∨
                                    last symbol = lst ∨
                                    last symbol = mor ∨
                                    last symbol = uqu
end  relatoperator last symbol;


Boolean procedure  booloperator last symbol;
begin   booloperator last symbol:= last symbol = qvl ∨
                                   last symbol = imp ∨
                                   last symbol = or  ∨
                                   last symbol = and
end  booloperator last symbol;


Boolean procedure  declarator last symbol;
begin   own type:= last symbol = own; if  own type then  next symbol;
        type:= if last symbol = rea   then 0 else
               if last symbol = integ then 1 else
               if last symbol = boole then 2 else
               if last symbol = stri  then 3 else 1000;
        if type < 4 then next symbol
                    else begin if own type then  ERRORMESSAGE (104);
                               if last symbol = array then  type:= 0
                         end;
        arr declarator last symbol:= last symbol = array;
        if arr declarator last symbol ∧ run number = 300
        then arr decla macro:= if own type
                               then (if type = 0 then ORAD else
                                     if type = 1 then OIAD else
                                     if type = 2 then OBAD else OSTAD)
                               else (if type = 0 then RAD else
                                     if type = 1 then IAD else
                                     if type = 2 then BAD else STAD);
        chara:= if arr declarator last symbol
                then 8
                else if last symbol = switch
                     then 14
                     else if last symbol = proced
                          then (if type < 4 then 16 else 24)
                          else type;
        type declarator last symbol:= chara < 4;
        if own type ∧ chara > 8 then ERRORMESSAGE (105);
        if type < 4 ∧ last symbol = switch then ERRORMESSAGE (106);
        if chara < 25 ∧ run number = 100
        then character:= ((if type declarator last symbol
                           then type
                           else if type < 4
                                then type + chara
                                else chara) +
                          (if own type then 32 else 0)) × d19;
        declarator last symbol:= chara < 25
end  declarator last symbol;
```

```
Boolean procedure  specifier last symbol;
begin   type:= if last symbol = rea   then 0 else
               if last symbol = integ then 1 else
               if last symbol = boole then 2 else
               if last symbol = stri  then 3 else
               if last symbol = array then 5 else 1000;
        if type < 4 then next symbol;
        chara:= if last symbol = label  then 6 else
                if last symbol = switch then 14 else 1000;
        if type + chara < 1000 then ERRORMESSAGE (107);
        chara:= if last symbol = array  then 8 else
                if last symbol = proced then (if type < 4 then 16
                                                          else 24)
                                             else chara;
        if chara < 25 then next symbol;
        if chara + type < 2000 ∧ run number = 100
        then begin value character:= (if chara > 8 then type else
                                      if chara = 6 then    6 else
                                      if  type = 5 then    8
                                      else type + chara) + 64;
                    character:= ((if type > 5
                                  then chara
                                  else (if type > 1 then type else 4) +
                                       (if chara < 1000 then chara
                                                        else 0))
                                 + 96) × d19
             end;
        specifier last symbol:= chara + type < 2000
end  specifier last symbol;


Boolean procedure  operator last symbol;
begin   operator last symbol:= arithoperator last symbol ∨
                               relatoperator last symbol ∨
                               booloperator  last symbol
end  operator last symbol;
```

```
procedure  unsigned number;
begin   integer  sign of exponent;
        if last symbol < 10
        then begin value of constant:= unsigned integer (0);
                   real number:= digit last symbol
             end
        else begin value of constant:= if last symbol = ten then 1
                                                             else 0;
                   real number:= true
             end;
        decimal exponent:= 0;
        if real number
        then begin
             next0: if last symbol < 10
                    then begin decimal exponent:= decimal exponent + 1;
                               next symbol; goto next0
                         end;
                    if last symbol = period
                    then begin next symbol;
                               value of constant:=
                               unsigned integer (value of constant);
                               decimal exponent:= decimal exponent -
                                                  decimal count;
                        next1: if last symbol < 10
                               then begin next symbol; goto next1 end
                         end;
                    if last symbol = ten
                    then begin next symbol; sign of exponent:= 1;
                               if last symbol = plus
                               then next symbol
                               else if last symbol = minus
                                    then begin next symbol;
                                               sign of exponent:= - 1
                                         end;
                               decimal exponent:= decimal exponent +
                                                  sign of exponent ×
                                                  unsigned integer (0);
                               if last symbol < 10
                               then begin  ERRORMESSAGE (108);
                                    next2: if next symbol < 9
                                           then goto next2
                                    end
                         end
             end;
        small:= value of constant < d15 ∧ ¬ real number
end  unsigned number;
```

```
integer procedure  unsigned integer (start); integer  start;
begin   integer  word;
        word:= start; decimal count:= 0;
        if last symbol > 9 then ERRORMESSAGE (109);
next0:  if last symbol < 10
        then begin if word < 6710886 ∨ (word = 6710886 ∧ last symbol < 4)
                   then begin word:= 10 × word + last symbol;
                              decimal count:= decimal count + 1;
                              next symbol; goto next0
                        end
             end;
        unsigned integer:= word
end  unsigned integer;


procedure  read identifier;
begin   integer  word, count;
        word:= count:= word count:= 0;
        if letter last symbol
        then
        begin
        next0: if last symbol < 64
               then
               begin if count = 4
                     then begin word count:= word count + 1;
                                word:= count:= 0
                          end;
                     word:= space[nl base - nlp - word count]:=
                     d6 × word - last symbol - 1;
                     count:= count + 1; next symbol; goto next0
               end
               else
               begin last identifier:= space[nl base - nlp];
                     last identifier1:= if word count = 0
                                        then 0
                                        else space[nl base - nlp - 1]
               end
        end
        else begin ERRORMESSAGE (110); space[nl base - nlp]:= - 1 end;
        space[nl base - nlp - word count - 1]:= 127 × d19
end  read identifier;


integer procedure  next pointer (n); integer  n;
begin   integer  word, pointer;
        pointer:= n;
next0:  word:= - space[nl base - pointer];
        if word <  0 then begin pointer:= pointer + 1; goto next0 end;
        if word ≥ d25 then begin pointer:= word - word : d13 × d13;
                                 goto next0
                           end;
        next pointer:= pointer
end  next pointer;
```

```
integer procedure  look up;
begin   integer  count, pointer;
        pointer:= block cell pointer +
                  (if in formal list ∨ in array declaration
                   then 5 else 4);
next0:  pointer:= next pointer (pointer);
        for count:= 0 step 1 until word count do
        begin if space[nl base - pointer - count] ≠
                 space[nl base - last nlp - count]
              then goto next1
        end;
        pointer:= pointer + word count + 1;
        if space[nl base - pointer] < 0
        then begin next1: pointer:= pointer + 1;
                          goto if space[nl base - pointer] < 0 then next1
                                                               else next0
             end;
        look up:= pointer
end  look up;


Boolean procedure  in name list;
begin   integer  head;
        if real number ∨ ¬ int labels
        then in name list:= false
        else begin head:= value of constant : d18;
                   space[nl base - nlp]:= - d12 - head;
                   space[nl base - nlp - 1]:=
                   (head - 1) × d18 - value of constant;
                   word count:= 1;
                   space[nl base - nlp - 2]:= 6 × d19;
                   last nlp:= nlp; integer label:= look up;
                   in name list:= integer label < nlp
             end
end  in name list;


integer procedure  next identifier (n); integer  n;
begin   integer  pointer;
        pointer:= next pointer (n) + 1;
next0:  if  space[nl base - pointer] < 0
        then begin pointer:= pointer + 1; goto next0 end;
        next identifier:= pointer
end  next identifier;
```

```
procedure  skip identifier;
begin   if last symbol < 64 then begin next symbol; skip identifier end
end  skip identifier;


procedure  skip type declaration;
begin   if letter last symbol then skip identifier;
        if last symbol = comma
        then begin next symbol; skip type declaration end
end  skip type declaration;


procedure  skip value list;
begin   if last symbol = value
        then begin next symbol; skip type declaration;
                   if last symbol = semicolon then next symbol
             end
end  skip value list;


procedure  skip specification list;
begin   if specifier last symbol
        then begin skip type declaration;
                   if last symbol = semicolon then next symbol;
                   skip specification list
             end
end  skip specification list;


procedure skip string;
begin   quote counter:= 1;
next0:  if next symbol ≠ unquote then goto next0;
        quote counter:= 0
end  skip string;


procedure skip rest of statement (pr); procedure pr;
begin   if last symbol = do
        then begin next symbol; pr end
        else
        if last symbol = goto ∨ last symbol = for ∨
           last symbol = begin
        then pr;
        if last symbol = quote then skip string;
        if last symbol ≠ semicolon ∧ last symbol ≠ end
        then begin next symbol;
                   skip rest of statement (pr)
             end
end  skip rest of statement;
```

```
integer procedure bit string (kn, n, code word); integer kn,n,code word;
begin   integer  k;
        k:= code word : kn; bit string:= (code word - k × kn) : n
end  bit string;


integer procedure  display level;
begin   display level:=
        bit string (d6, d0, space[nl base - block cell pointer - 1])
end  display level;


integer procedure  top of display;
begin   top of display:=
        bit string (d13, d6, space[nl base - block cell pointer - 1])
end  top of display;


integer procedure  local space;
begin   local space:= space[nl base - block cell pointer - 1] : d13
end  local space;


integer procedure  proc level;
begin   proc level:=
        bit string (d6, d0, space[nl base - block cell pointer - 2])
end  proc level;


Boolean procedure  use of counter stack;
begin   use of counter stack:=
        bit string (d7, d6, space[nl base - block cell pointer - 2]) = 1
end  use of counter stack;

integer procedure  status;
begin   status:= space[nl base - block cell pointer - 2] : d13
end status;


Boolean procedure  in code (n); integer  n;
begin   in code:= bit string (d25, d24, space[nl base - n - 1]) = 1
end  in code;


integer procedure  type bits (n); integer  n;
begin   type bits:= bit string (d22, d19, space[nl base - n])
end type bits;
```

```
Boolean procedure  local label (n); integer  n;
begin   local label:=
        nonformal label (n) ∧
        bit string(d6, d0,
        space[nl base - corresponding block cell pointer (n) - 1]) =
        display level
end  local label;


Boolean procedure  nonformal label (n); integer  n;
begin   nonformal label:= space[nl base - n] : d19 = 6
end nonformal label;


integer procedure  corresponding block cell pointer (n); integer  n;
begin   integer  p;
        p:= block cell pointer;
next0:  if n < p ∨ (n > space[nl base - p - 2] : d13 ∧ p > 0)
        then begin p:= space[nl base - p] : d13; goto next0 end;
        corresponding block cell pointer:= p
end  corresponding block cell pointer;


procedure  entrance block;
begin   block cell pointer:= next block cell pointer;
        next block cell pointer:=
        bit string (d13, d0, space[nl base - block cell pointer])
end  entrance block;


procedure  exit block;
begin   block cell pointer:= space[nl base - block cell pointer] : d13
end  exit block;


procedure  init;
begin   stock:= stock1:= last symbol:= word count:= - 1;
        shift:= 1;
        line counter:= quote counter:= forcount:= 0;
        in formal list:= in array declaration:= false;
        case:= lower case; text pointer:= 0
end init;
```

```
procedure test pointers;
begin   integer fprog, fnl, i, shift;
        if text in memory
        then
        begin fprog:= text base +
                      (if runnumber = 300 then text pointer else 0) -
                      instruct counter;
              fnl:= nl base - nlp -
                    (text base +
                     (if runnumber = 100 then text pointer
                                         else end of text));
              if fprog + fnl < 40
              then begin text in memory:= false; test pointers end
              else if fprog < 20
                   then begin shift:= (fnl - fprog) : 2;
                              for i:= text base + text pointer
                              step - 1 until text base do
                              space[i + shift]:= space[i];
                              text base:= text base + shift
                        end
                   else if fnl < 20
                        then
                        begin shift:= (fprog - fnl) : 2;
                              for i:= text base step 1
                              until text base + text pointer do
                              space[i]:= space[i + shift];
                              text base:= text base - shift
                        end
        end
        else if nl base - nlp - instruct counter < 20
             then begin ERRORMESSAGE (492); goto endrun end
end  test pointers;
```

```
procedure prescan0;
begin   integer old block cell pointer, displ level, prc level,
                global count, local count, label count, local for count,
                max for count, internal block depth, string occurrence,
                subcount, array pointer;


procedure  Program;
begin   integer  n;
        character:= 6 × d19;
        if letter last symbol
        then begin read identifier;
                   if last symbol = colon
                   then begin n:= Process identifier;
                              Label declaration (n)
                        end
                   else ERRORMESSAGE (111);
                   Program
             end
        else
        if digit last symbol
        then begin unsigned number;
                   if last symbol = colon then Int lab declaration
                                          else ERRORMESSAGE (112);
                   Program
             end
        else
        if last symbol = begin
        then Begin statement
        else begin ERRORMESSAGE (113); next symbol; Program end
end  Program;


integer procedure  Block (proc identifier); integer  proc identifier;
begin   integer  dump1, dump2, dump3, dump4, dump5, dump6, dump7, dump8,
                 n, formal count;
        dump1:= block cell pointer; dump2:= local for count;
        dump3:= max for count;      dump4:= local count;
        dump5:= label count;        dump6:= internal block depth;
        dump7:= string occurrence;  dump8:= prc level;
        local for count:= max for count:= local count:= label count:=
        internal block depth:= string occurrence:= 0;
        block cell pointer:= nlp + 1;
        space[nl base - old block cell pointer]:=
        space[nl base - old block cell pointer] + block cell pointer;
        old block cell pointer:= block cell pointer;
        space[nl base - block cell pointer]:= dump1 × d13;
        space[nl base - block cell pointer - 1]:= displ level:=
                                                  displ level + 1;
        space[nl base - block cell pointer - 3]:= 0;
        nlp:= nlp + 6;
```

```
if proc identifier > 0
then
begin prc level:= displ level; formal count:= 0;
      space[nl base - block cell pointer - 4]:= - d25 - nlp;
      if last symbol = open
      then begin  character:= 127 × d19;
           next0: next symbol; Identifier;
                  space[nl base - nlp]:= 0; nlp:= nlp + 1;
                  formal count:= formal count + 1;
                  if last symbol = comma then goto next0;
                  if last symbol = close then next symbol
                                         else ERRORMESSAGE (114)
           end;
      if last symbol = semicolon then next symbol
                                 else ERRORMESSAGE (115);
      space[nl base - proc identifier - 1]:=
                                 d22 + formal count + 1;
      if last symbol = value
      then
      begin
      next1: next symbol; n:= Identifier;
             if n > last nlp  then ERRORMESSAGE (116)
                              else space[nl base - n]:= 95 × d19;
             nlp:= last nlp;
             if last symbol = comma then goto next1;
             if last symbol = semicolon then next symbol
                                        else ERRORMESSAGE (117)
      end;
next2: if specifier last symbol
      then
      begin
      next3: n:= Identifier;
             if n > last nlp
             then ERRORMESSAGE (118)
             else if space[nl base - n] = 127 × d19
                  then space[nl base - n]:= character
                  else if space[nl base - n] ≠  95 × d19
                       then ERRORMESSAGE (119)
                       else if value character > 75
                            then ERRORMESSAGE (120)
                            else
                            begin space[nl base - n]:=
                                  value character × d19;
                                  if type = 3
                                  then string occurrence:= d6
                            end;
             nlp:= last nlp;
             if last symbol = comma
             then begin next symbol; goto next3 end;
             if last symbol = semicolon then next symbol
                                        else ERRORMESSAGE (121);
             goto next2
      end;
```

```
                space[nl base - nlp]:= - d25 - 4 - dump1; nlp:= nlp + 1;
                space[nl base - block cell pointer - 4]:= - d25 - nlp;
                if last symbol = quote
                then begin  space[nl base - proc identifier - 1]:=
                            space[nl base - proc identifier - 1] + d24;
                     next4: next symbol;
                            if last symbol ≠ unquote then goto next4;
                            next symbol
                     end
                else
                if last symbol = begin
                then begin next symbol;
                           if declarator last symbol then Declaration list;
                           Compound tail; next symbol
                     end
                else Statement
           end
           else
           begin space[nl base - nlp]:= - d25 - 4 - dump1; nlp:= nlp + 1;
                 space[nl base - block cell pointer - 4]:= - d25 - nlp;
                 Declaration list; Compound tail
           end;

           space[nl base - block cell pointer - 2]:=
           d13 × nlp + string occurrence + prc level;
           for n:= 0 step 1 until max for count - 1 do
           space[nl base - nlp - n]:= d19;
           space[nl base - block cell pointer - 1]:=
           space[nl base - block cell pointer - 1] +
           d6 × (internal block depth + 1);
           if prc level > 1
           then space[nl base - block cell pointer - 1]:=
                space[nl base - block cell pointer - 1] +
                d13 × (max for count + local count)
           else global count:= global count + max for count +
                               local count + label count;
           nlp:= nlp + max for count;
           space[nl base - nlp]:= - d25 - 5 - block cell pointer;
           nlp:= nlp + 1;
           space[nl base - block cell pointer + 1]:= - d25 - nlp;
           displ level:= space[nl base - dump1 - 1];
           Block:= internal block depth + 1;
           block cell pointer:= dump1; local for count:= dump2;
           max for count:= dump3;      local count:= dump4;
           label count:= dump5;        internal block depth:= dump6;
           string occurrence:= dump7;  prc level:= dump8
end  Block;


procedure  Compound tail;
begin   Statement; if  last symbol = semicolon
                   then begin next symbol; Compound tail end
end  Compound tail;
```

```
procedure  Declaration list;
begin   integer  n, count;
next0:  if type declarator last symbol
        then begin  count:= 0;
             next1: count:= count + 1;
                    n:= Identifier;
                    if n < last nlp then ERRORMESSAGE (122);
                    if last symbol = comma
                    then begin next symbol; goto next1 end;
                    if type = 0 ∨ type = 3 then count:= 2 × count;
                    if own type then global count:= global count + count
                                else local count:= local count + count;
                    if type = 3 then string occurrence:= d6
             end
        else
        if arr declarator last symbol
        then begin  count:= array pointer:= 0;
             next2: count:= count + 1;
                    next symbol; n:= Identifier;
                    if n < last nlp then ERRORMESSAGE (123);
                    space[nl base - nlp]:= array pointer;
                    array pointer:= nlp; nlp:= nlp + 1;
                    if last symbol = comma then goto next2;
                    dimension:= 0;
                    if last symbol = sub
                    then
                    begin  subcount:= 1;
                    next3: next symbol;
                           if letter last symbol
                           then skip identifier
                           else if digit last symbol
                                then begin unsigned number;
                                           Store numerical constant
                                     end;
                           if last symbol = quote then skip string;
                           if last symbol = colon
                           then begin dimension:= dimension + 1;
                                      goto next3
                                end;
                           if last symbol = sub
                           then begin subcount:= subcount + 1;
                                      goto next3
                                end;
                           if last symbol ≠ bus then goto next3;
                           if subcount > 1
                           then begin subcount:= subcount - 1;
                                      goto next3
                                end;
                           next symbol;
                           if dimension = 0 then ERRORMESSAGE (124)
                           else dimension:= dimension + 1
                    end
                    else ERRORMESSAGE (125);
```

```
            next4: n:= space[nl base - array pointer];
                   space[nl base - array pointer]:= dimension;
                   array pointer:= n;
                   if n ≠ 0 then goto next4;
                   if own type
                   then global count:=
                        global count + (3 × dimension + 3) × count
                   else local count:= local count + count;
                   if last symbol = comma
                   then begin count:= 0; goto next2 end;
                   if type = 3 then string occurrence:= d6
            end
        else
        if last symbol = switch
        then begin  next symbol; n:= Identifier;
                   if n < last nlp then ERRORMESSAGE (126);
                   space[nl base - nlp]:= 0; nlp:= nlp + 1;
            next5: next symbol;
                   if letter last symbol
                   then skip identifier
                   else if digit last symbol
                        then begin unsigned number;
                                   Store numerical constant
                             end;
                   if last symbol = quote then skip string;
                   if last symbol ≠ semicolon then goto next5
            end
        else begin  next symbol; n:= Identifier;
                   if n < last nlp then ERRORMESSAGE (127);
                   nlp:= nlp + 1;
                   if type < 4
                   then begin space[nl base - nlp]:= type × d19;
                              nlp:= nlp + 1
                        end;
                   Block (n)
            end;
        if last symbol = semicolon then next symbol
                                   else ERRORMESSAGE (128);
        if declarator last symbol then goto next0
end  Declaration list;
```

```
procedure  Statement;
begin   integer  n, lfc;
        lfc:= local for count;
next0:  character:= 6 × d19;
next1:  if  letter last symbol
        then begin read identifier;
                   if last symbol = colon
                   then begin n:= Process identifier;
                              Label declaration (n);
                              goto next1
                        end
             end
        else
        if digit last symbol
        then begin unsigned number;
                   if last symbol = colon
                   then begin Int lab declaration; goto next1 end
                   else Store numerical constant
             end
        else
        if last symbol = for
        then begin local for count:= local for count + 1;
                   if local for count > max for count
                   then max for count:= local for count
             end
        else
        if last symbol = begin
        then begin Begin statement; next symbol; goto next1 end
        else
        if last symbol = quote then skip string;
        if last symbol ≠ semicolon ∧ last symbol ≠ end
        then begin next symbol; goto next1 end;
        local for count:= lfc
end  Statement;


procedure  Label declaration (n); integer  n;
begin   if n < last nlp then ERRORMESSAGE (129);
        if label count = 0
        then space[nl base - block cell pointer - 3]:= d13 × (nlp - 1);
        label count:= label count + 2;
        space[nl base - nlp]:= d18; nlp:= nlp + 1;
        next symbol
end  Label declaration;


procedure  Int lab declaration;
begin   if real number
        then begin ERRORMESSAGE (130); next symbol end
        else begin int labels:= true;
                   in name list; nlp:= nlp + 3;
                   Label declaration (integer label)
             end
end  Int lab declaration;
```

```
procedure  Begin statement;
begin    integer  n;
         next symbol;
         if declarator last symbol
         then begin n:= Block (0);
                    if n > internal block depth
                    then internal block depth:= n
              end
         else Compound tail
end  Begin statement;


procedure  Store numerical constant;
begin    if ¬ small
         then begin space[prog base + instruct counter]:=
                    value of constant;
                    space[prog base + instruct counter + 1]:=
                    decimal exponent;
                    instruct counter:= instruct counter + 2
              end
end  Store numerical constant;


integer procedure  Process identifier;
begin    last nlp:= nlp; nlp:= nlp + word count + 2;
         space[nl base - nlp + 1]:= character;
         Process identifier:= look up
end  Process identifier;


integer procedure  Identifier;
begin    read identifier;
         Identifier:= Process identifier
end  Identifier;
```

```
main program of prescan0:
        runnumber:= 100; init;
        local for count:= max for count:= local count:= label count:=
        global count:= internal block depth:= string occurrence:=
        displ level:= prc level:= 0;
        old block cell pointer:= block cell pointer:= nlp;
        int labels:= false;
        space[text base]:=
        space[nl base - block cell pointer]:=
        space[nl base - block cell pointer - 1]:=
        space[nl base - block cell pointer - 3]:= 0;
        nlp:= block cell pointer + 6;
        space[nl base - block cell pointer - 4]:= - d25 - nlp;
        next symbol;
        Program;
        space[nl base - block cell pointer - 1]:=
                (global count + max for count + label count) × d13 +
                (internal block depth + 1) × (d13 + d6);
        space[nl base - block cell pointer - 2]:= nlp × d13;
        for n:= 0 step 1 until max for count - 1 do
        space[nl base - nlp - n]:= d19;
        nlp:= nlp + max for count;
        space[nl base - block cell pointer - 5]:= - d25 - nlp;
        end of text:= text pointer;
        output
end  prescan0;
```

```
procedure prescan1;
begin


procedure  Arithexp;
begin   if last symbol = if then Ifclause (Arithexp)
                          else Simple arithexp
end  Arithexp;


procedure  Simple arithexp;
begin   integer  n;
        if last symbol = plus ∨ last symbol = minus
        then
next0:  next symbol;
        if last symbol = open
        then begin next symbol; Arithexp;
                   if last symbol = close then next symbol
             end
        else
        if digit last symbol then unsigned number
        else
        if letter last symbol
        then begin n:= Identifier; Arithmetic (n);
                   Subscripted variable(n); Function designator(n)
             end
        else
        if last symbol = if then Arithexp;
        if arithoperator last symbol then goto next0
end  Simple arithexp;


procedure  Subscripted variable (n); integer  n;
begin   if last symbol = sub then begin Subscrvar (n);
                                        dimension:= Subscrlist;
                                        List length (n)
                                  end
end  Subscripted variable;


integer procedure  Subscrlist;
begin   next symbol; Arithexp;
        if last symbol = comma then Subscrlist:= Subscrlist + 1
                               else begin if last symbol = bus
                                          then next symbol;
                                          Subscrlist:= 1
                                    end
end  Subscrlist;


procedure  Boolexp;
begin   if last symbol = if then Ifclause (Boolexp)
                          else Simple boolean
end  Boolexp;
```

```
procedure  Simple boolean;
begin   integer  n, type;
        if last symbol = non  then next symbol;
        if last symbol = open then begin next symbol; Exp (type);
                                         if last symbol = close
                                         then next symbol
                                   end
        else
        if letter last symbol then begin n:= Identifier;
                                         Subscripted variable (n);
                                         Function designator (n);
                                         if arithoperator last symbol V
                                            relatoperator last symbol
                                         then Arithmetic (n)
                                         else Boolean (n)
                                   end
        else
        if digit last symbol V last symbol = plus V last symbol = minus
        then Simple arithexp
        else
        if last symbol = true V last symbol = false then next symbol;
        Rest of exp (type)
end  Simple boolean;


procedure  Stringexp;
begin   if last symbol = if then Ifclause (Stringexp)
                         else Simple stringexp
end  Stringexp;


procedure  Simple stringexp;
begin   integer  n;
        if last symbol = open
        then begin next symbol; Stringexp;
                   if last symbol = close then next symbol
             end
        else
        if letter last symbol
        then begin n:= Identifier; String (n);
                   Subscripted variable (n);
                   Function designator (n)
             end
        else
        if last symbol = quote
        then begin  quote counter:= 1;
             next0: next symbol;
                    if last symbol = unquote
                    then begin quote counter:= 0;
                               next symbol
                         end
                    else goto next0
             end
end  Simple stringexp;
```

```
procedure  Desigexp;
begin   if last symbol = if then Ifclause (Desigexp)
                          else Simple desigexp
end  Desigexp;


procedure  Simple desigexp;
begin   integer  n;
        if last symbol = open
        then begin next symbol; Desigexp;
                   if last symbol = close then next symbol
             end
        else
        if letter last symbol
        then begin n:= Identifier; Designational (n);
                   Subscripted variable (n)
             end
        else
        if digit last symbol
        then begin unsigned number;
                   if in name list
                   then Designational (integer label)
             end
end  Simple desigexp;


procedure  Exp (type); integer  type;
begin   if last symbol = if
        then begin next symbol; Boolexp;
                   next symbol; Simplexp (type);
                   if last symbol = else
                   then begin next symbol; Type exp (type) end
             end
        else Simplexp (type)
end  Exp;


procedure  Type exp (type); integer  type;
begin   if type = ar ∨ type = re ∨ type = in
        then Arithexp
        else if type = bo
             then Boolexp
             else if type = st
                  then Stringexp
                  else if type = des
                       then Desigexp
                       else Exp (type)
end  Type exp;
```

```
procedure  Simplexp (type); integer  type;
begin   integer  n;
        type:= un;
        if last symbol = open
        then begin next symbol; Exp (type);
                   if last symbol = close then next symbol
             end
        else
        if letter last symbol
        then begin n:= Identifier; Subscripted variable (n);
                   Function designator (n);
                   if arithoperator last symbol ∨
                      relatoperator last symbol
                   then Arithmetic (n)
                   else if booloperator last symbol
                        then Boolean (n)
                        else begin if nonformal label (n)
                                   then Designational (n);
                                   type:= type bits (n)
                             end
             end
        else
        if digit last symbol
        then begin unsigned number;
                   if in name list
                   then Designational (integer label)
                   else  type:= ar
             end
        else
        if last symbol = plus ∨ last symbol = minus
        then begin Simple arithexp; type:= ar end
        else
        if last symbol = non ∨ last symbol = true ∨ last symbol = false
        then begin Simple boolean; type:= bo end
        else
        if last symbol = quote
        then begin Simple stringexp; type:= st; goto end end;
        Rest of exp (type);
end:
end  Simplexp;


procedure  Rest of exp (type); integer  type;
begin   if arithoperator last symbol
        then begin next symbol; Simple arithexp;
                   type:= ar
             end;
        if relatoperator last symbol
        then begin next symbol; Simple arithexp;
                   type:= bo
             end;
        if booloperator last symbol
        then begin next symbol; Simple boolean;
                   type:= bo
             end
end  Rest of exp;
```

```
procedure  Assignstat (n); integer  n;
begin   Subscripted variable (n);
        if  last symbol = colonequal then  Right hand side (n)
end  Assignstat;


procedure  Right hand side (n); integer  n;
begin   integer  m, type, type n;
        Assigned to (n); type n:= type bits (n);
        next symbol;
        if letter last symbol
        then begin m:= Identifier; Subscripted variable (m);
                   if  last symbol = colonequal
                   then
                   begin Insert (type n, m);
                         Right hand side (m); type:= type bits (m)
                   end
                   else
                   begin Function designator (m);
                         if arithoperator last symbol ∨
                            relatoperator last symbol
                         then Arithmetic (m)
                         else if booloperator last symbol
                              then Boolean (m)
                              else
                              begin Arbost (m);
                                    type:= if type n = re ∨ type n = in
                                           then ar
                                           else type n;
                                    Insert (type, m);
                                    type:= type bits (m);
                                    if type = re ∨ type = in
                                    then type:= ar
                              end;
                         Rest of exp (type)
                   end
             end
        else begin m:= type n; Type exp (type n);
                   if m ≠ nondes then type n:= m;
                   type:= if type n = re ∨ type n = in then ar
                                                      else type n
             end;
        Insert (type, n)
end  Right hand side;
```

```
procedure  Insert (type, n); integer  type, n;
begin   if type = re
        then Real (n)
        else if type = in
             then Integer (n)
             else if type = bo
                  then Boolean (n)
                  else if type = ar then Arithmetic (n)
end  Insert;


procedure  Function designator (n); integer  n;
begin   if last symbol = open then begin Function (n);
                                         dimension:= Parlist;
                                         List length (n)
                                   end
end  Function designator;


integer procedure  Parlist;
begin   next symbol; Actual parameter;
        if last symbol = comma
        then Parlist:= Parlist + 1
        else begin if last symbol = close then next symbol;
                   Parlist:= 1
             end
end  Parlist;


procedure  Actual parameter;
begin   integer  type;
        Exp (type)
end  Actual parameter;


procedure  Procstat (n); integer  n;
begin   Proc (n);
        dimension:= if last symbol = open then Parlist else 0;
        List length (n)
end  Procstat;
```

```
procedure  Statement;
begin   integer  n;
        if letter last symbol
        then begin n:= Identifier;
                   if last symbol = colon
                   then Labelled statement (n)
                   else begin if last symbol = sub ∨
                                 last symbol = colonequal
                              then Assignstat (n)
                              else Procstat (n)
                        end
             end
        else
        if digit last symbol
        then begin unsigned number;
                   if last symbol = colon
                   then  Intlabelled statement
             end
        else
        if last symbol = goto then Gotostat
        else
        if last symbol = begin
        then begin next symbol;
                   if declarator last symbol then Block
                                             else Compound tail;
                   next symbol
             end
        else
        if last symbol = if then Ifclause (Statement)
        else
        if last symbol = for then Forstat
end  Statement;


procedure  Gotostat;
begin   integer  n;
        next symbol;
        if letter last symbol
        then begin n:= Identifier;
                   if ¬ local label (n)
                   then begin Designational (n);
                              Subscripted variable (n)
                        end
             end
        else Desigexp
end  Gotostat;


procedure  Compound tail;
begin   Statement;
        if last symbol ≠ semicolon ∧ last symbol ≠ end
        then skip rest of statement (Statement);
        if last symbol = semicolon
        then begin next symbol; Compound tail end
end  Compound tail;
```

```
procedure  Ifclause (pr); procedure  pr;
begin   next symbol; Boolexp;
        if last symbol = then then next symbol;
        pr;
        if last symbol = else then begin next symbol; pr end
end  Ifclause;


procedure  Forstat;
begin   integer  n;
        next symbol;
        if letter last symbol
        then begin  n:= Identifier; Arithmetic (n);
                    Subscripted variable (n);
                    if last symbol = colonequal
                    then
             next0: next symbol; Arithexp;
                    if last symbol = step
                    then begin next symbol; Arithexp;
                               if last symbol = until
                               then begin next symbol;
                                          Arithexp
                                    end
                         end
                    else
                    if last symbol = while
                    then begin next symbol; Boolexp end;
                    if last symbol = comma then goto next0;
                    if last symbol = do    then next symbol;
                    forcount:= forcount + 1;
                    Statement;
                    forcount:= forcount - 1
             end
end  Forstat;


procedure  Switch declaration;
begin   integer  n;
        next symbol;
        if letter last symbol
        then begin n:= Identifier;
                   if last symbol = colonequal
                   then begin dimension:= Switchlist;
                              Switch length (n)
                        end
             end
end  Switch declaration;


integer procedure  Switchlist;
begin   next symbol; Desigexp;
        if last symbol = comma then Switchlist:= Switchlist + 1
                               else Switchlist:= 1
end  Switchlist;
```

```
procedure  Array declaration;
begin   integer  i, n, count;
        next symbol; n:= Identifier; count:= 1;
next0:  if last symbol = comma then begin next symbol;
                                          if letter last symbol
                                          then skip identifier;
                                          count:= count + 1; goto next0
                                     end;
        if last symbol = sub   then begin in array declaration:= true;
                                          dimension:= Bound pair list;
                                          in array declaration:= false
                                     end
                               else dimension:= 0;
        Check dimension (n);
        if own type then for i:= 1 step 1 until count do
                         begin Address (n, instruct counter);
                               instruct counter:= instruct counter +
                                                  3 × dimension + 6;
                               n:= next identifier (n)
                         end;
        if last symbol = comma then Array declaration
end  Array declaration;


integer procedure  Bound pair list;
begin   next symbol; Arithexp;
        if last symbol = colon then begin next symbol; Arithexp end;
        if last symbol = comma
        then Bound pair list:= Bound pair list + 1
        else begin if last symbol = bus then next symbol;
                   Bound pair list:= 1
             end
end  Bound pair list;
```

```
procedure  Procedure declaration;
begin   integer  n, m;
        next symbol; n:= Identifier; entrance block;
        if last symbol = open
        then begin  in formal list:= true ;
             next0: next symbol; m:= Identifier;
                    if space[nl base - m] = 95 × d19
                    then begin ERRORMESSAGE (201);
                               space[nl base - m]:= 127 × d19
                         end;
                    if last symbol = comma then goto next0;
                    if last symbol = close then next symbol;
                    in formal list:= false
             end;
        if last symbol = semicolon then next symbol;
        skip value list; skip specification list;
        if in code (n)
        then Scan code (n)
        else begin if space[nl base - n] : d19 = 19 ∧
                   ¬ use of counter stack
                   then space[nl base - block cell pointer - 2]:=
                        space[nl base - block cell pointer - 2] + 64;
                   if last symbol = begin
                   then begin next symbol;
                              if declarator last symbol
                              then Declaration list;
                              Compound tail; next symbol
                         end
                   else Statement;
                   Addressing of block identifiers (n)
             end
end  Procedure declaration;


procedure  Block;
begin   entrance block; Declaration list; Compound tail;
        Addressing of block identifiers (0)
end  Block;


procedure  Declaration list;
begin   if typedeclarator last symbol then skip type declaration
        else
        if arr declarator last symbol then Array declaration
        else
        if last symbol = switch       then Switch declaration
                                      else Procedure declaration;
        if last symbol = semicolon    then next symbol;
        if declarator last symbol     then Declaration list
end  Declaration list;
```

```
procedure  Program;
begin   integer  n;
        if letter last symbol
        then begin n:= Identifier;
                   if last symbol = colon
                   then Label declaration (n);
                   Program
             end
        else
        if digit last symbol
        then begin unsigned number;
                   if in name list
                   then Label declaration (integer label);
                   Program
             end
        else
        if last symbol = begin
        then begin next symbol;
                   if declarator last symbol
                   then Block
                   else Compound tail
             end
        else begin next symbol; Program end
end  Program;


procedure  Labelled statement (n); integer  n;
begin   if nonformal label (n) then Label declaration (n);
        Statement
end  Labelled statement;


procedure  Intlabelled statement;
begin   if in name list then Label declaration (integer label);
        Statement
end  Intlabelled statement;


procedure  Label declaration (n); integer  n;
begin   if proc level = 0
        then begin Designational (n); Address (n, instruct counter);
                   space[nl base - n - 1]:=
                   space[nl base - n - 1] + instruct counter +
                   d20 × forcount;
                   space[prog base + instruct counter]:= 0;
                   space[prog base + instruct counter + 1]:=
                   d18 × display level + dp0;
                   instruct counter:= instruct counter + 2
             end
        else space[nl base - n - 1]:= space[nl base - n - 1] +
                                      d20 × forcount;
        next symbol
end  Label declaration;
```

```
procedure  Addressing of block identifiers (n); integer  n;
begin   integer  counter, f, code, code1;
        if n = 0 then space[nl base - block cell pointer - 1]:=
                      space[nl base - block cell pointer - 1] + d13;
        if proc level > 0
        then
        begin counter:= d9 × display level + d8;
              if n = 0
              then counter:= counter + 1 + d18
              else
              begin counter:= counter + display level + top of display;
                    f:= block cell pointer + 5;
              next0: f:= next identifier (f);
                    if f > block cell pointer
                    then
                    begin Address (f, counter);
                          code1:= space[nl base - f] : d18;
                          code:= code1 : 2;
                          counter:= counter +
                          (if code = 64 ∨ code = 67 ∨ code = 70
                           then 2
                           else if code < 96
                                then 1
                                else if code1 = 2 × code
                                     then 2 else 4);
                          goto next0
                    end;
                    counter:= counter + d18;
                    code:= space[nl base - n] : d19;
                    if code ≠ 24
                    then
                    begin f:= if wanted then 3 else
                              if code = 16 ∨ code = 19 then 2 else 1;
                          Address (n + 2, counter);
                          counter:= counter + f;
                          space[nl base - block cell pointer - 1]:=
                          space[nl base - block cell pointer - 1] +
                          d13 × f
                    end
              end;
```

```
            f:= status;
      next1: if space[nl base - f] > 0
            then begin Address (f, counter); counter:= counter + 1;
                       f:= f + 1;
                       goto next1
                 end;
            f:= block cell pointer + 4;
      next2: f:= next identifier (f); code:= space[nl base - f] : d19;
            if f > block cell pointer ∧ f < status ∧ code < 64
            then begin if code > 24
                       then
                       begin if  code < 36
                             then
                             begin Address (f, instruct counter);
                                   instruct counter:=
                                   instruct counter +
                                   (if code = 32 ∨ code = 35
                                    then 2 else 1)
                             end
                       end
                       else
                       if code < 14
                       then
                       begin if code ≠ 6 ∨
                                (code = 6 ∧
                                 bit string (d19, d18,
                                 space[nl base - f - 1]) = 0)
                             then
                             begin Address (f, counter);
                                   counter:=
                                   counter +
                                   (if code = 0 ∨ code = 3 ∨ code = 6
                                    then  2 else  1)
                             end
                       end;
                       goto next2
                 end;
            if counter > d18 + d9 × (display level + 1)
            then ERRORMESSAGE (202);
            exit block
        end
        else Static addressing
end  Addressing of block identifiers;
```

```
procedure  Static addressing;
begin   integer  f, code;
        f:= status;
next0:  if space[nl base - f] > 0
        then begin Address (f, instruct counter);
                   instruct counter:= instruct counter + 1; f:= f + 1;
                   goto next0
             end;
        f:= block cell pointer + 4;
next1:  f:= next identifier (f); code:= space[nl base - f] : d19;
        if f > block cell pointer ∧ f < status
        then begin if code > 24 ∧ code < 36 ∨ code < 14 ∧ code ≠ 6
                   then begin Address (f, instruct counter);
                              instruct counter:=
                              instruct counter +
                              (if code = 0 ∨ code = 3 ∨
                                  code = 32 ∨ code = 35 then 2
                                                        else 1)
                        end;
                   goto next1
             end;
        exit block
end Static addressing;
```

```
procedure  Add type (n, t); integer  n, t;
begin   integer  code, new code, type;
        new code:= code:= space[nl base - n] : d19;
        if code > 95
        then begin if code = 127
                   then new code:= 96 + t
                   else if code = 120 ∧ t < 6
                        then new code:= 112 + t
                        else
                        begin type:= code - code : 8 × 8;
                              if type = un ∨ (type = nondes ∧ t < 5) ∨
                                 (type = ar ∧ t < 2)
                              then new code:= code - type + t
                        end;
                   space[nl base - n]:=
                   space[nl base - n] - (code - new code) × d19
             end
end  Add type;

procedure  Real (n); integer  n;  begin Add type (n, re) end Real;

procedure  Integer (n); integer  n;  begin Add type (n, in) end Integer;

procedure  Boolean (n); integer  n;  begin Add type (n, bo) end Boolean;

procedure  String (n); integer  n;  begin Add type (n, st) end String;

procedure  Arithmetic (n); integer  n;
begin  Add type (n, ar) end  Arithmetic;

procedure  Arbost (n); integer  n;
begin   Add type (n, nondes) end  Arbost;
```

```
procedure  Designational (n); integer  n;
begin   integer  p;
        if nonformal label (n)
        then
        begin if bit string (d19, d18, space[nl base - n - 1]) = 1
              then
              begin space[nl base - n - 1]:=
                    abs (space[nl base - n - 1] - d18);
                    p:= corresponding block cell pointer (n);
                    if bit string (d6, d0, space[nl base - p - 2]) > 0
                    then begin space[nl base - p - 3]:=
                                space[nl base - p - 3] + 1;
                                space[nl base - p - 1]:=
                                space[nl base - p - 1] + d14
                         end
              end
        end
        else Add type (n, des)
end  Designational;


procedure  Assigned to (n); integer  n;
begin   integer  code;
        code:= space[nl base - n] : d19;
        if code > 95
        then
        begin if code = 127 then code:= 101;
              if code < 102 then space[nl base - n]:= code × d19 + d18
                            else Add type (n, nondes)
        end
end Assigned to;


procedure  Subscrvar (n); integer  n;
begin   integer  code, new code;
        code:= space[nl base - n] : d19;
        if code > 95
        then begin new code:= if code = 127
                              then 111
                              else if code < 104
                                   then code + 8
                                   else code;
                   space[nl base - n]:= space[nl base - n] +
                                        (new code - code) × d19
             end
end  Subscrvar;
```

```
procedure  Proc (n); integer  n;
begin   integer  code, new code;
        code:= space[nl base - n] : d19;
        if code > 95
        then begin new code:= if code = 127
                              then 120
                              else if code < 102
                                   then code + 16
                                   else code;
                   space[nl base - n]:= space[nl base - n] +
                                        (new code- code) × d19
             end
end  Proc;

procedure  Function (n); integer  n;
begin   Arbost (n); Proc (n) end  Function;

procedure  List length (n); integer  n;
begin   integer  word;
        if space[nl base - n] : d19 > 95
        then begin word:= space[nl base - n - 1];
                   if bit string (d18, d0, word) = 0
                   then space[nl base - n - 1]:= word + dimension + 1
             end
end  List length;

procedure  Switch length (n);  integer  n;
begin   space[nl base - n - 1]:= dimension + 1 end  Switch length;

procedure  Address (n, m); integer  n, m;
begin   integer  word;
        word:= space[nl base - n] : d18;
        space[nl base - n]:= word × d18 + m
end  Address;

procedure  Check dimension (n); integer  n;
begin   if space[nl base - n - 1] ≠ dimension + 1
        then begin ERRORMESSAGE (203);
                   space[nl base - n - 1]:= dimension + 1
             end
end  Check dimension;
```

```
integer procedure  Identifier;
begin   integer  n;
        last nlp:= nlp; read identifier; Identifier:= n:= look up;
        if n > nlp then Ask librarian;
        if n > nlp then begin ERRORMESSAGE (204);
                              nlp:= nlp + word count + 3;
                              space[nl base - nlp + 1]:= 0
                        end
end  Identifier;


procedure  Scan code (n); integer  n;
begin   block cell pointer:= space[nl base - block cell pointer] : d13;
next0:  next symbol; if  last symbol = minus then next symbol;
        if letter last symbol then Identifier else unsigned integer (0);
        if last symbol = comma then goto next0;
        if last symbol = unquote then next symbol
end  Scan code;


procedure  Ask librarian;
begin   comment if the current identifier occurs in the library
                then this procedure will add a new namecell to
                the name list and increase nlp;
end  Ask librarian;
```

```
main program of prescan1:
        if ¬ text in memory
        then begin NEWPAGE;
                   PRINTTEXT (*input tape for prescan1*)
             end;
        runnumber:= 200; init;
        block cell pointer:= next block cell pointer:= 0;
        dp0:= instruct counter;
        instruct counter:= instruct counter + top of display;
        space[nl base - nlp]:= - 1;
        next symbol; entrance block;
        Program; Static addressing;
        output
end  prescan1;
```

```
procedure translate;
begin

integer last lnc, lnc, last lncr, macro, parameter, state,
        stack0, stack1, b, ret level, max depth,
        ret max depth, max depth isr, max display length,
        max proc level, ecount, controlled variable, increment,
        10, 11, 12, 13, 14, 15, number of switch elements,
        switch identifier, switch list count, sword,
        address of constant, sum of maxima;

Boolean in switch declaration, in code body, if statement forbidden,
        complicated, complex step element;


procedure  Arithexp;
begin   integer  future1, future2;
        if last symbol = if
        then begin  future1:= future2:= 0;
                    next symbol; Boolexp; Macro2 (COJU, future1);
                    if last symbol ≠ then then ERRORMESSAGE (300)
                                       else next symbol;
                    Simple arithexp;
                    if last symbol = else
                    then begin  Macro2 (JU, future2);
                                Substitute (future1);
                                next symbol; Arithexp;
                                Substitute (future2)
                         end
                    else ERRORMESSAGE (301)
             end
        else Simple arithexp
end  Arithexp;


procedure  Simple arithexp;
begin   if last symbol = minus then begin  next symbol; Term;
                                            Macro (NEG)
                                    end
                               else begin  if last symbol = plus
                                           then next symbol;
                                           Term
                                    end;
        Next term
end  Simple arithexp;
```

```
procedure  Next term;
begin   if last symbol = plus  then begin  Macro (STACK);
                                            next symbol; Term;
                                            Macro (ADD); Next term
                                       end
        else
        if last symbol = minus then begin  Macro (STACK);
                                            next symbol; Term;
                                            Macro (SUB); Next term
                                       end
end  Next term;

procedure  Term;  begin  Factor; Next factor end  Term;

procedure  Next factor;
begin   if last symbol = mul then begin  Macro (STACK);
                                          next symbol; Factor;
                                          Macro (MUL); Next factor
                                     end
        else
        if last symbol = div then begin  Macro (STACK);
                                          next symbol; Factor;
                                          Macro (DIV); Next factor
                                     end
        else
        if last symbol = idi then begin  Macro (STACK);
                                          next symbol; Factor;
                                          Macro (IDI); Next factor
                                     end
end  Next factor;

procedure  Factor;  begin  Primary; Next primary end  Factor;

procedure  Next primary;
begin   if last symbol = ttp then begin  Macro (STACK);
                                          next symbol; Primary;
                                          Macro (TTP); Next primary
                                     end
end  Next primary;
```

```
procedure  Primary;
begin   integer  n;
        if last symbol = open then begin  next symbol; Arithexp;
                                          if last symbol = close
                                          then next symbol
                                          else ERRORMESSAGE (302)
                                  end
        else
        if digit last symbol  then begin  Unsigned number;
                                          Arithconstant
                                  end
        else
        if letter last symbol then begin  n:= Identifier;
                                          Subscripted variable (n);
                                          Function designator (n);
                                          Arithname (n)
                                  end
        else
        begin  ERRORMESSAGE (303);
               if last symbol = if ∨ last symbol = plus ∨
                  last symbol = minus
               then Arithexp
        end
end  Primary;


procedure  Arithname (n); integer n;
begin   if Nonarithmetic (n) then  ERRORMESSAGE (304);
        complicated:= Formal (n) ∨ Function (n);
        if Simple (n)
        then begin  if Formal (n)  then  Macro2 (DOS, n) else
                    if Integer (n) then  Macro2 (TIV, n)
                                   else  Macro2 (TRV, n)
             end
end  Arithname;


procedure  Subscripted variable (n); integer  n;
begin   if Subscrvar (n) then begin  Address description (n);
                                     if last symbol = colonequal
                                     then begin  Macro (STACK);
                                                 Macro (STAA)
                                          end
                                     else Evaluation of (n)
                              end
end  Subscripted variable;
```

```
procedure  Address description (n); integer  n;
begin   if last symbol = sub
        then begin  next symbol; dimension:= Subscript list;
                    Check dimension (n);
                    if Formal (n)        then  Macro2 (DOS, n) else
                    if Designational (n) then  Macro2 (TSWE, n)
                                         else  Macro2 (TAK, n)
             end
        else ERRORMESSAGE (305)
end  Address description;


procedure  Evaluation of (n); integer  n;
begin   if  Designational(n)
        then begin  if Formal (n) then  Macro (TFSL)
                                  else  Macro (TSL)
             end
        else
        if  Boolean (n) then Macro (TSB)  else
        if  String (n)  then Macro (TSST) else
        if  Formal (n)  then Macro (TFSU) else
        if  Integer (n) then Macro (TSI)  else Macro (TSR)
end  Evaluation of;


integer procedure  Subscript list;
begin   Arithexp;
        if last symbol = comma
        then begin  Macro (STACK); next symbol;
                    Subscript list:= Subscript list + 1
             end
        else begin  if last symbol = bus
                    then next symbol
                    else ERRORMESSAGE (306);
                    Subscript list:= 1
             end
end  Subscript list;
```

```
procedure  Boolexp;
begin   integer future1, future2;
        if  last symbol = if
        then begin  future1:= future2:= 0;
                    next symbol; Boolexp; Macro2 (COJU, future1);
                    if last symbol ≠ then then ERRORMESSAGE (307)
                                       else next symbol;
                    Simple boolean;
                    if last symbol = else
                    then begin Macro2 (JU, future2);
                               Substitute (future1);
                               next symbol; Boolexp;
                               Substitute (future2)
                         end
                    else ERRORMESSAGE (308)
             end
        else Simple boolean
end Boolexp;

procedure  Simple boolean;
begin   Implication; Next implication end  Simple boolean;

procedure  Next implication;
begin   if last symbol = qvl then begin  Macro (STAB);
                                          next symbol; Implication;
                                          Macro (QVL); Next implication
                                  end
end  Next implication;

procedure  Implication;  begin  Boolterm; Next boolterm end  Implication;

procedure  Next boolterm;
begin   if last symbol = imp then begin  Macro (STAB);
                                          next symbol; Boolterm;
                                          Macro (IMP); Next boolterm
                                  end
end  Next boolterm;

procedure  Boolterm;  begin  Boolfac; Next boolfac end  Boolterm;

procedure  Next boolfac;
begin   if last symbol = or  then begin  Macro (STAB);
                                          next symbol; Boolfac;
                                          Macro (OR); Next boolfac
                                  end
end  Next boolfac;

procedure  Boolfac;  begin  Boolsec; Next boolsec end  Boolfac;
```

```
procedure  Next boolsec;
begin   if last symbol = and then begin  Macro (STAB);
                                          next symbol; Boolsec;
                                          Macro (AND); Next boolsec
                                  end
end  Next boolsec;


procedure  Boolsec;
begin   if last symbol = non then begin  next symbol; Boolprim;
                                          Macro (NON)
                                  end
                           else Boolprim
end  Boolsec;


procedure  Boolprim;
begin   integer type, n;
        if last symbol = open
        then begin next symbol; Arboolexp (type);
                   if last symbol = close then next symbol
                                          else ERRORMESSAGE (309);
                   if type = ar   then Rest of relation else
                   if type = arbo then begin if arithoperator last symbol
                                             then Rest of relation
                                             else Relation
                                       end
             end
        else
        if letter last symbol then begin n:= Identifier;
                                         Subscripted variable (n);
                                         Boolprimrest (n)
                                   end
        else
        if digit last symbol ∨ last symbol = plus ∨ last symbol = minus
        then begin Simple arithexp; Rest of relation end
        else
        if last symbol = true ∨ last symbol = false
        then begin Macro2 (TBC, last symbol); next symbol end
        else ERRORMESSAGE (310)
end  Boolprim;
```

```
Boolean procedure  Relation;
begin   integer relmacro;
        if  relatoperator last symbol
        then begin  relmacro:= Relatmacro; Macro (STACK);
                    next symbol; Simple arithexp;
                    Macro (relmacro); Relation:= true
             end
        else Relation:= false
end  Relation;


procedure  Rest of relation;
begin   Rest of arithexp;
        if ¬ Relation then  ERRORMESSAGE (311)
end  Rest of relation;


procedure  Boolprimrest (n); integer  n;
begin   Function designator (n);
        if Arithmetic (n) ∨ arithoperator last symbol
           ∨ relatoperator last symbol
        then begin  Arithname (n); Rest of relation end
        else Boolname (n)
end  Boolprimrest;


procedure  Boolname (n); integer  n;
begin   if Nonboolean (n) then ERRORMESSAGE (312);
        if Simple (n)     then begin if Formal (n) then  Macro2 (DOS, n)
                                                   else  Macro2 (TBV, n)
                                   end
end  Boolname;


procedure  Arboolexp (type); integer  type;
begin   integer  future1, future2;
        if last symbol = if
        then begin  future1:= future2:= 0;
                    next symbol; Boolexp; Macro2 (COJU, future1);
                    if last symbol ≠ then then ERRORMESSAGE (313)
                                       else next symbol;
                    Simple arboolexp (type);
                    if last symbol = else
                    then
                    begin  Macro2 (JU, future2); Substitute (future1);
                           next symbol;
                           if type = bo then Boolexp else
                           if type = ar then Arithexp
                                        else Arboolexp (type);
                           Substitute (future2)
                    end
                    else ERRORMESSAGE (314)
             end
        else Simple arboolexp (type)
end  Arboolexp;
```

```
procedure  Simple arboolexp (type); integer  type;
begin   integer  n;
        if last symbol = open
        then begin  next symbol; Arboolexp (type);
                    if last symbol = close then next symbol
                                           else ERRORMESSAGE (315);
                    if type = bo ∨
                       type = arbo ∧ booloperator last symbol
                    then begin Rest of boolexp; type:= bo end
                    else if type = ar ∨
                               arithoperator last symbol ∨
                               relatoperator last symbol
                         then Rest of arboolexp (type)
             end
        else
        if letter last symbol
        then begin  n:= Identifier; Subscripted variable (n);
                    Arboolrest (type, n)
             end
        else
        if digit last symbol ∨ last symbol = plus ∨ last symbol = minus
        then begin Simple arithexp; Rest of arboolexp (type) end
        else
        if last symbol = non ∨ last symbol = true ∨ last symbol = false
        then begin Simple boolean; type:= bo end
        else
        begin  ERRORMESSAGE (316); type:= arbo end
end  Simple arboolexp;


procedure  Rest of arithexp;
begin   Next primary; Next factor; Next term end  Rest of arithexp;


procedure  Rest of boolexp;
begin   Next boolsec; Next boolfac; Next boolterm; Next implication
end Rest of boolexp;


procedure  Rest of arboolexp (type); integer  type;
begin   Rest of arithexp;
        if Relation
        then begin  Rest of boolexp; type:= bo end else type:= ar
end  Rest of arboolexp;
```

```
procedure  Arboolrest (type, n); integer  type, n;
begin   Function designator (n);
        if Boolean (n) ∨ booloperator last symbol
        then begin  Boolname (n); Rest of boolexp; type:= bo end
        else
        if  Arithmetic (n) ∨ arithoperator last symbol ∨
            relatoperator last symbol
        then begin  Arithname (n); Rest of arboolexp (type) end
        else begin  if String (n) ∨ Designational (n)
                    then ERRORMESSAGE (317);
                    Macro2 (DOS, n); type:= arbo
             end
end  Arboolrest;


procedure  Stringexp;
begin   integer  future1, future2;
        if last symbol = if
        then begin  future1:= future2:= 0;
                    next symbol; Boolexp; Macro2 (COJU, future1);
                    if last symbol ≠ then then ERRORMESSAGE (318)
                                       else next symbol;
                    Simple stringexp;
                    if last symbol = else
                    then begin  Macro2 (JU, future2);
                                Substitute (future1);
                                next symbol; Stringexp;
                                Substitute (future2)
                         end
                    else ERRORMESSAGE (319)
             end
        else Simple stringexp
end  Stringexp;


procedure  Simple stringexp;
begin   integer  future, n;
        if last symbol = open
        then begin  next symbol; Stringexp;
                    if last symbol = close then next symbol
                                       else ERRORMESSAGE (320)
             end
        else
        if  letter last symbol
        then begin  n:= Identifier; Subscripted variable (n);
                    Stringname (n)
             end
        else
        if last symbol = quote
        then begin  Macro (TCST); future:= 0; Macro2 (JU, future);
                    Constant string; Substitute (future)
             end
        else ERRORMESSAGE (321)
end  Simple stringexp;
```

```
procedure  Stringname (n); integer  n;
begin   if Nonstring (n) then  ERRORMESSAGE (322);
        Function designator (n);
        if Simple (n) then begin  if Formal (n) then  Macro2 (DOS, n)
                                                else  Macro2 (TSTV, n)
                               end
end  Stringname;


procedure  Desigexp;
begin   integer  future1, future2;
        if last symbol = if
        then begin  future1:= future2:= 0;
                    next symbol; Boolexp; Macro2 (COJU, future1);
                    if last symbol ≠ then then ERRORMESSAGE (323)
                                       else next symbol;
                    Simple desigexp;
                    if last symbol = else
                    then begin  Macro2 (JU, future2);
                                Substitute (future1);
                                next symbol; Desigexp;
                                Substitute (future2)
                         end
                    else ERRORMESSAGE (324)
             end
        else Simple desigexp
end  Desigexp;


procedure  Simple desigexp;
begin   integer  n;
        if  last symbol = open
        then begin  next symbol; Desigexp;
                    if last symbol = close then next symbol
                                           else ERRORMESSAGE (325)
             end
        else
        if letter last symbol then begin n:= Identifier;
                                         Subscripted variable (n);
                                         Designame (n)
                                   end
        else
        if  digit last symbol then begin Unsigned number;
                                         if in name list
                                         then Macro2 (TLV, integer label)
                                         else ERRORMESSAGE (326)
                                   end
        else ERRORMESSAGE (327)
end  Simple desigexp;
```

```
procedure  Designame (n); integer  n;
begin   if Nondesignational (n) then ERRORMESSAGE (328);
        if Simple (n)
        then begin  if Formal (n) then Macro2 (DOS, n)
                                  else Macro2 (TLV, n)
             end
end  Designame;


procedure  Ardesexp (type); integer  type;
begin   Exp (type);
        if type = bo ∨ type = st then ERRORMESSAGE (329);
        if type = un             then type:= intlab else
        if type = nondes         then type:= ar
end  Ardesexp;


procedure  Nondesexp (type); integer  type;
begin   Exp (type);
        if type = des    then ERRORMESSAGE (330);
        if type = un     then type:= nondes else
        if type = intlab then type:= ar
end  Nondesexp;


procedure  Exp (type); integer  type;
begin   integer  future1, future2;
        if last symbol = if
        then begin  future1:= future2:= 0;
                    next symbol; Boolexp; Macro2 (COJU, future1);
                    if last symbol ≠ then then ERRORMESSAGE (331)
                                       else next symbol;
                    Simplexp (type);
                    if last symbol = else
                    then
                    begin  Macro2 (JU, future2);
                           Substitute (future1); next symbol;
                           if type = ar     then Arithexp        else
                           if type = bo     then Boolexp         else
                           if type = st     then Stringexp       else
                           if type = des    then Desigexp        else
                           if type = intlab then Ardesexp (type) else
                           if type = nondes then Nondesexp (type)
                                            else Exp (type);
                           Substitute (future2)
                    end
                    else  ERRORMESSAGE (332)
             end
        else  Simplexp (type)
end  Exp;
```

```
procedure  Simplexp (type); integer  type;
begin   integer  n;
        if last symbol = open
        then begin  next symbol; Exp (type);
                    if last symbol = close then next symbol
                                           else ERRORMESSAGE (333);
                    if type = bo ∨ (type = nondes ∨ type = un) ∧
                       booloperator last symbol
                    then begin Rest of boolexp; type:= bo end
                    else
                    if type ≠ st ∧ type ≠ des ∧ operator last symbol
                    then  Rest of arboolexp (type)
             end
        else
        if letter last symbol
        then begin  n:= Identifier; Subscripted variable (n);
                    Exprest (type, n)
             end
        else
        if digit last symbol
        then begin  Unsigned number; Arithconstant;
                    if in name list ∧ ( ¬ operator last symbol)
                    then begin  Macro2 (TLV, integer label);
                                type:= intlab
                         end
                    else Rest of arboolexp (type)
             end
        else
        if last symbol = plus ∨ last symbol = minus
        then  Simple arboolexp (type)
        else
        if last symbol = non ∨ last symbol = true ∨ last symbol = false
        then begin  Simple boolean; type:= bo end
        else
        if last symbol = quote then begin Simple stringexp; type:= st end
        else
        begin  ERRORMESSAGE (334); type:= un end
end  Simplexp;
```

```
procedure  Exprest (type, n); integer  type, n;
begin   if Designational (n) then begin Designame (n); type:= des end
        else
        if String (n)        then begin Stringname (n); type:= st end
        else
        begin  Function designator (n);
               if Boolean (n) ∨ booloperator last symbol
               then begin Boolname (n); Rest of boolexp; type:= bo end
               else
               if Arithmetic (n) ∨ arithoperator last symbol ∨
                  relatoperator last symbol
               then begin Arithname (n); Rest of arboolexp (type) end
               else begin if Simple (n) then  Macro2 (DOS, n);
                          type:= if Unknown (n) then un else nondes
                    end
        end
end  Exprest;


procedure  Assignstat (n); integer  n;
begin   Subscripted variable (n);
        if last symbol = colonequal then Distribute on type (n)
                                    else ERRORMESSAGE (335)
end  Assignstat;


integer procedure  Distribute on type (n); integer  n;
begin   if Integer (n)
        then begin  Intassign (n); Distribute on type:= in end
        else
        if  Real (n)
        then begin  Realassign (n); Distribute on type:= re end
        else
        if Boolean (n)
        then begin  Boolassign (n); Distribute on type:= bo end
        else
        if String (n)
        then begin  Stringassign (n); Distribute on type:= st end
        else Distribute on type:= if Arithmetic (n) then Arassign (n)
                                                    else Unassign (n)
end  Distribute on type;


procedure  Prepare (n); integer  n;
begin   if  Function (n)
        then begin  if Formal (n) then ERRORMESSAGE (336)
                    else
                    if  Outside declaration (n) then ERRORMESSAGE (337)
                    else n:= Local position (n)
             end
        else if Simple (n) ∧ Formal (n) then Macro2 (DOS2, n);
        next symbol
end  Prepare;
```

```
Boolean procedure  Intassign (n); integer  n;
begin   integer m;  Boolean  rounded;
        if  Noninteger (n) then  ERRORMESSAGE (338);
        Prepare (n); rounded:= false;
        if letter last symbol
        then begin  m:= Identifier; Subscripted variable (m);
                    if last symbol = colonequal
                    then rounded:= Intassign (m)
                    else begin  Function designator (m);
                                Arithname (m); Rest of arithexp
                         end
             end
        else Arithexp;
        if Subscrvar (n)
        then begin  if Formal (n) then Macro (STFSU)
                    else
                    if rounded    then Macro (SSTSI)
                                  else Macro (STSI)
             end
        else if Formal (n) then Macro2 (DOS3, n)
        else if rounded    then Macro2 (SSTI, n)
                           else Macro2 (STI, n);
        Intassign:= Formal (n) ] rounded
end  Intassign;


procedure  Realassign (n); integer  n;
begin   integer  m;
        if Nonreal (n) then ERRORMESSAGE (339);
        Prepare (n);
        if letter last symbol
        then begin m:= Identifier; Subscripted variable (m);
                   if last symbol = colonequal
                   then Realassign (m)
                   else begin Function designator (m);
                              Arithname (m); Rest of arithexp
                        end
             end
        else Arithexp;
        if  Subscrvar (n)
        then begin if Formal (n) then Macro (STFSU)
                                 else Macro (STSR)
             end
        else if Formal (n) then Macro2 (DOS3, n)
                           else Macro2 (STR, n)
end  Realassign;
```

```
procedure  Boolassign (n); integer  n;
begin   integer  m;
        if Nonboolean (n) then  ERRORMESSAGE (340);
        Prepare (n);
        if letter last symbol
        then begin m:= Identifier; Subscripted variable (m);
                   if last symbol = colonequal
                   then Boolassign (m)
                   else begin Boolprimrest (m); Rest of boolexp end
             end
        else Boolexp;
        if  Subscrvar (n) then Macro (STSB)
                          else if Formal (n) then Macro2 (DOS3, n)
                                             else Macro2 (STB, n)
end  Boolassign;


procedure  Stringassign (n); integer  n;
begin   integer  m;
        if  Nonstring (n) then  ERRORMESSAGE (341);
        Prepare (n);
        if letter last symbol
        then begin m:= Identifier; Subscripted variable (m);
                   if last symbol = colonequal
                   then Stringassign (m)
                   else Stringname (m)
             end
        else Stringexp;
        if Subscrvar (n) then Macro (STSST)
                         else if Formal (n) then Macro2 (DOS3, n)
                                            else Macro2 (STST, n)
end  Stringassign;


integer procedure  Arassign (n); integer  n;
begin   integer  type, m;
        if  Nonarithmetic (n) then  ERRORMESSAGE (342);
        Prepare (n); type:= ar;
        if letter last symbol
        then begin m:= Identifier; Subscripted variable (m);
                   if last symbol = colonequal
                   then begin if Nonarithmetic (m)
                              then ERRORMESSAGE (343);
                              type:= Distribute on type (m)
                        end
                   else begin Function designator (m);
                              Arithname (m); Rest of arithexp
                        end
             end
        else Arithexp;
        if Subscrvar (n) then Macro (STFSU) else Macro2 (DOS3, n);
        Arassign:= type
end  Arassign;
```

```
integer procedure  Unassign (n); integer  n;
begin   integer  type, m;
        if  Nontype (n) then  ERRORMESSAGE (344);
        Prepare (n);
        if letter last symbol
        then begin m:= Identifier; Subscripted variable (m);
                   if Nontype (m) then ERRORMESSAGE (345);
                   if last symbol = colonequal
                   then type:= Distribute on type (m)
                   else Exprest (type, m)
             end
        else Nondesexp (type);
        if Subscrvar (n)
        then begin if type = bo then  Macro (STSB)
                   else
                   if type = st then  Macro (STSST)
                                else  Macro (STFSU)
             end
        else Macro2 (DOS3, n);
        Unassign:= type
end  Unassign;


procedure  Function designator (n); integer  n;
begin   if Proc (n)
        then begin if Nonfunction (n) then ERRORMESSAGE (346);
                   Procedure call (n)
             end
end  Function designator;


procedure  Procstat (n); integer  n;
begin   if  Proc (n)
        then begin Procedure call (n);
                   if  ¬ (In library (n) ∨  Function (n))
                   then last lnc:= - n;
                   if  Formal (n) ∨ (Function (n) ∧ String (n))
                   then  Macro (REJST)
             end
        else ERRORMESSAGE (347)
end  Procstat;
```

```
procedure  Procedure call (n); integer  n;
begin   integer  number of parameters;
        if Operator like (n)
        then Process operator (n)
        else begin number of parameters:= List length (n);
                   if number of parameters ≠ 0
                   then Parameter list (n, number of parameters)
                   else if Formal (n)
                        then Macro2 (DOS, n)
                        else if In library(n) then Macro2 (ISUBJ, n)
                                              else Macro2 (SUBJ, n)
             end
end  Procedure call;


integer procedure  Ordinal number (n); integer  n;
begin   Ordinal number:= if Formal (n) then 15
                         else
                         if Subscrvar (n)
                         then (if Arithmetic (n)
                               then (if Real (n) then 8 else 9)
                               else if Boolean (n)
                                    then 10 else 11)
                         else
                         if Function (n)
                         then (if  Arithmetic (n)
                               then (if  Real (n)  then 24 else 25)
                               else if Boolean (n) then 26 else 27)
                         else
                         if Proc (n) then 30
                         else
                         if Arithmetic(n)
                         then (if Real (n) then 0 else 1)
                         else if Boolean (n)
                              then 2
                              else if String (n) then 3 else 14
end  Ordinal number;
```

```
procedure  Parameter list (n, number of parameters);
integer  n, number of parameters;
begin integer  count, m, f, apd, type, future;
      Boolean  simple identifier;
      integer  array  descriptor list[1 : number of parameters];
      count:= future:= 0; f:= n;
      if last symbol = open
      then
      begin
      next: count:= count + 1; next symbol;
           Actual parameter (apd, simple identifier, type, future);
           if count ≤ number of parameters
           then
           begin descriptor list[count]:= apd;
                 if  ¬ Formal (n)
                 then
                 begin f:= Next formal identifier (f);
                       if simple identifier
                       then
                       begin if Subscrvar (f)
                             then
                             begin if Nonsubscrvar (type)
                                   then  ERRORMESSAGE (348);
                                   Check type (f, type);
                                   Check list length (f, type)
                             end
                             else
                             if Proc (f)
                             then
                             begin if Nonproc (type)
                                   then ERRORMESSAGE (349);
                                   Check list length (f, type);
                                   if Function (f)
                                   then begin if Nonfunction (type)
                                              then ERRORMESSAGE (350);
                                              Check type (f, type)
                                        end
                             end
                             else
                             if Simple (f)
                             then
                             begin if Nonsimple (type)
                                   then ERRORMESSAGE (351);
                                   Check type (f, type)
                             end
                       end
```

```
                    else
                    begin if Subscrvar.(f) ∨ Proc (f)
                          then ERRORMESSAGE (352);
                          if Assigned to (f) ∧ Nonassignable (apd)
                          then ERRORMESSAGE (353);
                          if Arithmetic(f) ∧
                             (type = bo ∨ type = st ∨ type = des)
                          then ERRORMESSAGE (354) else
                          if Boolean (f) ∧
                             type ≠ bo ∧ type ≠ nondes ∧ type ≠ un
                          then ERRORMESSAGE (355) else
                          if String (f) ∧
                             type ≠ st ∧ type ≠ nondes ∧ type ≠ un
                          then ERRORMESSAGE (356) else
                          if Designational (f) ∧
                             type ≠ des ∧ type ≠ un
                          then ERRORMESSAGE (357) else
                          if Arbost (f) ∧ type = des
                          then ERRORMESSAGE (358)
                    end
               end
          end
          else ERRORMESSAGE (359);
          if last symbol = comma then goto next;
          if last symbol = close
          then begin next symbol;
                     if count < number of parameters
                     then ERRORMESSAGE (360)
               end
          else ERRORMESSAGE (361)
     end
     else ERRORMESSAGE (362);
     if future ≠ 0 then  Substitute (future);
     if Formal (n) then  Macro2 (DOS, n) else  if In library (n)
                                               then Macro2 (ISUBJ, n)
                                               else Macro2 (SUBJ, n);
     m:= 0;
next apd: if m < count ∧ m < number of parameters
     then begin m:= m + 1; apd:= descriptor list[m];
                Macro2 (CODE, apd); goto next apd
          end
end  Parameter list;
```

```
procedure  Actual parameter (apd, simple identifier, type, future);
integer  apd, type, future; Boolean  simple identifier;
begin integer  n, begin address;
      begin address:= Order counter + (if future = 0 then 1 else 0);
      simple identifier:= false;
      if letter last symbol
      then
      begin n:= Identifier;
            if last symbol = comma ∨ last symbol = close
            then
            begin type:= n; simple identifier:= true;
                  if Proc (n) ∧ ¬ Formal (n)
                  then
                  begin if  future = 0 then Macro2 (JU, future);
                        Macro (TFD);
                        if In library (n) then Macro2 (IJU1, n)
                                          else Macro2 (JU1, n);
                        apd:= d20 × Ordinal number (n) + begin address
                  end
                  else if Subscrvar (n) ∧ Designational (n) ∧
                        ¬ Formal (n)
                       then begin if future = 0
                                  then Macro2 (JU, future);
                                  Macro2 (TSWE, n);
                                  apd:= 12 × d20 + begin address
                            end
                       else apd:= d20 × Ordinal number (n) +
                                  Address (n) +
                                  (if  Dynamic (n) then d18 else 0)
            end
            else
            begin Start implicit subroutine (future);
                  if Subscrvar (n) then Address description (n);
                  if (last symbol = comma ∨ last symbol = close) ∧
                     ( ¬ Designational (n))
                  then
                  begin if Unknown (n) then Macro (SAS);
                        Macro2 (EXITSV, - 2 × dimension);
                        apd:= d20 × (if Boolean (n) then 18 else
                                     if String (n)  then 19 else
                                     if Formal (n)  then 32 else
                                     if Real (n)    then 16 else 17)
                              + Order counter;
                        type:= if Arithmetic (n) then ar else
                               if Boolean (n)    then bo else
                               if String (n)     then st else
                               if Arbost (n)     then nondes else  un;
                        Macro2 (SUBJ, - begin address);
                        if Boolean (n) then  Macro (TASB)  else
                        if String (n)  then  Macro (TASST) else
                        if Formal (n)  then  Macro (TASU)  else
                        if Integer (n) then  Macro (TASI)
                                       else  Macro (TASR);
                        Macro (DECS); Macro2 (SUBJ, - begin address);
                        Macro (FAD)
                  end
```

```
                else
                begin if Subscrvar (n) then Evaluation of (n);
                      Exprest (type, n); Macro (EXITIS);
                      apd:= mask[type] + begin address
                end
        end
end
else
if digit last symbol
then begin Unsigned number;
           if (last symbol = comma ∨ last symbol = close) ∧
              ( ¬ in name list)
           then begin type:= ar; apd:= Number descriptor end
           else begin Start implicit subroutine (future);
                      Arithconstant;
                      if in name list ∧ ( ¬ operator last symbol)
                      then begin Macro2 (TLV, integer label);
                                 type:= intlab
                           end
                      else Rest of arboolexp (type);
                      Macro (EXITIS);
                      apd:= mask[type] + begin address
                end
     end
else
if last symbol = plus
then
begin next symbol;
      if digit last symbol
      then begin Unsigned number;
                 if last symbol = comma ∨ last symbol = close
                 then begin type:= ar; apd:= Number descriptor end
                 else begin Start implicit subroutine (future);
                            Arithconstant;
                            Rest of arboolexp (type);
                            Macro (EXITIS);
                            apd:= mask[type] + begin address
                      end
           end
      else begin Start implicit subroutine (future);
                 Arboolexp (type);
                 Macro (EXITIS); apd:= mask[type] + begin address
           end
end
```

```
      else
      if last symbol = minus
      then
      begin next symbol;
            if digit last symbol
            then begin Unsigned number;
                       if (last symbol = comma ∨ last symbol = close) ∧
                          small
                       then
                       begin type:= ar;
                             apd:= d20 × 13 + value of constant
                       end
                       else
                       begin Start implicit subroutine (future);
                             Arithconstant; Next primary; Next factor;
                             Macro (NEG); Rest of arboolexp (type);
                             Macro (EXITIS);
                             apd:= mask[type] + begin address
                       end
                 end
            else begin Start implicit subroutine (future);
                       Term; Macro (NEG);
                       Rest of arboolexp (type);
                       Macro (EXITIS); apd:= mask[type] + begin address
                 end
      end
      else
      if last symbol = true ∨ last symbol = false
      then
      begin type:= bo; n:= last symbol; next symbol;
            if last symbol = comma ∨ last symbol = close
            then apd:= d20 × 6 + (if n = true then 0 else 1)
            else begin Start implicit subroutine (future);
                       Macro2 (TBC, n);
                       Rest of boolexp;
                       Macro (EXITIS);
                       apd:= mask[type] + begin address
                 end
      end
      else begin Start implicit subroutine (future); Exp (type);
                 Macro (EXITIS); apd:= mask[type] + begin address
           end
end  Actual parameter;
```

```
procedure  Start implicit subroutine (future); integer  future;
begin   if future = 0 then  Macro2 (JU, future);
        Macro (ENTRIS)
end  Start implicit subroutine;


integer procedure  Number descriptor;
begin   Number descriptor:=
        if small then d20 × 7 + value of constant
                 else d20 × (if  real number then 4 else 5)
                      + address of constant
end  Number descriptor;


procedure  Process operator (n); integer  n;
begin   integer  count;
        count:= 0;
        if last symbol = open
        then begin
             next: next symbol; Arithexp; count:= count + 1;
                   if last symbol = comma
                   then begin Macro (STACK); goto next end;
                   if last symbol = close
                   then next symbol
                   else ERRORMESSAGE (361)
             end;
        if count ≠ List length (n) then ERRORMESSAGE (363);
        Macro (Operator macro (n))
end  Process operator;


Boolean procedure  Nonassignable (apd); integer  apd;
begin   integer  rank;
        rank:= apd : d20;
        Nonassignable:= (rank ≠ 15) ∧ (rank - rank : 16 × 16) > 3
end  Nonassignable;


procedure  Line;
begin   if lnc ≠ last lnc then Line1 end  Line;


procedure  Line1;
begin   if wanted then begin last lnc:= lnc; Macro2 (LNC, lnc) end
end  Line1;
```

```
procedure  Statement;
begin   ifstatement forbidden:= false; Stat end  Statement;


procedure  Unconditional statement;
begin   ifstatement forbidden:= true; Stat end  Unconditional statement;


procedure  Stat;
begin   integer  n, save lnc;
        if letter last symbol
        then begin save lnc:= line counter;
                   n:= Identifier;
                   if Designational (n)
                   then begin Label declaration (n); Stat end
                   else begin lnc:= save lnc; Line;
                              if Subscrvar (n) ∨ last symbol = colonequal
                              then Assignstat (n)
                              else Procstat (n)
                        end
             end
        else
        if digit last symbol
        then begin Unsigned number;
                   if in name list
                   then begin Label declaration (integer label); Stat end
                   else ERRORMESSAGE (364)
             end
        else begin if last symbol = goto
                   then begin lnc:= line counter; Line; Gotostat end
                   else
                   if last symbol = begin
                   then begin save lnc:= line counter; next symbol;
                              if declarator last symbol
                              then begin lnc:= save lnc; Line; Block end
                              else Compound tail;
                              next symbol
                        end
                   else
                   if last symbol = if
                   then begin if ifstatement forbidden
                              then ERRORMESSAGE (365);
                              lnc:= line counter; Line; Ifstat
                        end
                   else
                   if last symbol = for
                   then begin lnc:= line counter; Line; Forstat;
                              if last symbol = else
                              then ERRORMESSAGE (366)
                        end
             end
end  Stat;
```

```
procedure  Gotostat;
begin   integer  n;
        next symbol;
        if letter last symbol
        then begin n:= Identifier; Subscripted variable (n);
                   if local label (n)
                   then begin Test forcount (n); Macro2 (JU, n) end
                   else begin Designame (n); Macro (JUA) end
             end
        else begin Desigexp; Macro (JUA) end
end  Gotostat;


procedure  Compound tail;
begin   Statement;
        if last symbol ≠ semicolon ∧ last symbol ≠ end
        then begin ERRORMESSAGE (367);
                   skip rest of statement (Statement)
             end;
        if last symbol = semicolon
        then begin next symbol; Compound tail end
end  Compound tail;


procedure  Ifstat;
begin   integer  future1, future2, save lnc, last lnc1;
        future1:= future2:= 0; save lnc:= line counter;
        next symbol; Boolexp; Macro2 (COJU, future1);
        if last symbol = then then next symbol else ERRORMESSAGE (368);
        Unconditional statement;
        if last symbol = else
        then begin Macro2 (JU, future2); Substitute (future1);
                   last lnc1:= last lnc; last lnc:= save lnc;
                   next symbol; Statement; Substitute (future2);
                   if last lnc > last lnc1 then last lnc:= last lnc1
             end
        else begin Substitute (future1);
                   if last lnc > save lnc then last lnc:= save lnc
             end
end  Ifstat;


procedure  Forstat;
begin   integer  future, save lnc;
        save lnc:= line counter;
        10:= 0; next symbol; For list;
        future:= 0; Macro2 (JU, future); if 10 ≠ 0 then Substitute(10);
        if last symbol = do then next symbol else ERRORMESSAGE (369);
        Increase status (increment); forcount:= forcount + 1;
        Statement;
        Increase status (- increment); forcount:= forcount - 1;
        if last lnc < 0 ∨ lnc ≠ save lnc
        then begin lnc:= save lnc; Line1 end;
        Macro2 (IJU, status); Substitute (future)
end  Forstat;
```

```
procedure  Store preparation;
begin   if  Subscrvar (controlled variable) then Macro2 (SUBJ, - 12)
        else
        if  Formal (controlled variable)
        then Macro2 (DOS2, controlled variable)
end  Store preparation;


procedure  Store macro;
begin   if  Subscrvar (controlled variable)
        then begin if Formal (controlled variable)  then Macro (STFSU)
                   else
                   if Integer (controlled variable) then Macro (STSI)
                                                    else Macro (STSR);
                   Macro2 (DECB, 2)
             end
        else if Formal (controlled variable)
             then Macro2 (DOS3, controlled variable)
             else if Integer (controlled variable)
                  then Macro2 (STI, controlled variable)
                  else Macro2 (STR, controlled variable)
end  Store macro;


procedure  Take macro;
begin   if  Subscrvar (controlled variable)
        then Macro2 (SUBJ, - 11)
        else Arithname (controlled variable)
end  Take macro;


procedure  For list;
begin if letter last symbol
      then
      begin controlled variable:= Identifier;
            if Nonarithmetic (controlled variable)
            then ERRORMESSAGE (370);
            if Subscrvar (controlled variable)
            then
            begin 13:= 0; Macro2 (JU, 13);
                  14:= Order counter;
                  Address description (controlled variable);
                  Macro2 (EXITSV, 1 - 2 × dimension);
                  11:= Order counter;
                  Macro2 (SUBJ, - 14);
                  if Formal (controlled variable)  then Macro (TSCVU)
                  else
                  if Integer (controlled variable) then Macro (TISCV)
                                                   else Macro (TRSCV);
                  12:= Order counter;
                  Macro2 (SUBJ, - 14); Macro (FADCV);
                  Substitute (13)
            end
            else if Function (controlled variable)
                 then ERRORMESSAGE (371);
            if last symbol ≠ colonequal then ERRORMESSAGE (372);
```

```
list: 13:= Order counter;
      Macro2 (TSIC, 0); Macro2 (SSTI, status);
      14:= Order counter;
      Store preparation;
      next symbol; Arithexp;
      if last symbol = comma ∨ last symbol = do
      then begin Store macro; Macro2 (JU, 10);
                 Substitute (13)
           end
      else
      if last symbol = while
      then begin Store macro;
                 next symbol; Boolexp;
                 Macro2 (YCOJU, 10); Subst2 (14, 13)
           end
      else
      if last symbol = step
      then begin 15:= 0; Macro2 (JU, 15); 14:= Order counter;
                 next symbol; complicated:= false; Arithexp;
                 complex step element:=
                 complicated ∨ Order counter > 14 + 1;
                 if complex step element then  Macro (EXIT);
                 Substitute (13);
                 Store preparation; Take macro; Macro (STACK);
                 if complex step element then  Macro2 (SUBJ, - 14)
                                       else  Macro2 (DO, 14);
                 Macro (ADD);
                 Substitute (15);
                 Store macro;
                 if Subscrvar (controlled variable) ∨
                    Formal (controlled variable)
                 then Take macro;
                 Macro (STACK);
                 if last symbol = until
                 then begin next symbol; Arithexp end
                 else ERRORMESSAGE (373);
                 Macro (TEST1);
                 if complex step element then Macro2 (SUBJ, - 14)
                                       else Macro2 (DO, 14);
                 Macro (TEST2); Macro2 (YCOJU, 10)
           end
      else ERRORMESSAGE (374);
      if last symbol = comma then goto list
 end
 else ERRORMESSAGE (375)
end  For list;
```

```
procedure  Switch declaration;
begin integer  m;
      next symbol;
      if letter last symbol
      then
      begin switch identifier:= Identifier;
            number of switch elements:= List length (switch identifier);
            if last symbol = colonequal
            then
            begin integer array
                  sword list[1 : number of switch elements];
                  switch list count:= 0; in switch declaration:= true;
            next: switch list count:= switch list count + 1;
                  next symbol;
                  if letter last symbol
                  then
                  begin m:= Identifier;
                        if Nondesignational (m) then ERRORMESSAGE (376);
                        if Subscrvar (m)
                        then
                        begin sword:= - 45613055 + Order counter;
                              Subscripted variable (m); Macro (EXIT)
                        end
                        else
                        sword:= (if Formal (m)
                                 then -33685503
                                 else 4718592 + (if Dynamic (m)
                                                 then function digit
                                                 else 0)) +
                                    Address (m)
                  end
                  else
                  if digit last symbol
                  then
                  begin Unsigned number;
                        if  in name list
                        then  sword:= 4718592 +
                                      (if Dynamic (integer label)
                                       then function digit
                                       else 0) +
                                      Address (integer label)
                        else ERRORMESSAGE (377)
                  end
                  else
                  begin sword:= - 45613055 + Order counter;
                        Desigexp; Macro (EXIT)
                  end;
```

```
                if switch list count > number of switch elements
                then ERRORMESSAGE (378);
                sword list[switch list count]:= sword;
                if last symbol = comma then goto next;
                if switch list count < number of switch elements
                then ERRORMESSAGE (379);
                Mark position in name list (switch identifier);
                in switch declaration:= false;
                Macro2 (CODE, number of switch elements);
                m:= 0;
          next sword: if m < switch list count ∧
                         m < number of switch elements
                then begin m:= m + 1; sword:= sword list[m];
                           Macro2 (CODE, sword); goto next sword
                     end
          end
          else ERRORMESSAGE (380)
     end
     else ERRORMESSAGE (381)
end  Switch declaration;


procedure  Array declaration;
begin   integer  n, count;
        next symbol; lnc:= line counter; Line;
        n:= Identifier; dimension:= List length (n); count:= 1;
next:   if last symbol = comma then begin next symbol; Identifier;
                                           count:= count + 1; goto next
                                    end;
        if  last symbol = sub  then begin in array declaration:= true;
                                          Bound pair list;
                                          in array declaration:= false
                                    end
                               else ERRORMESSAGE (382);
        Macro2 (TNA, count); Macro2 (TDA, dimension);
        Macro2 (TAA, n); Macro (arr decla macro);
        if last symbol = comma then Array declaration
end  Array declaration;


procedure  Bound pair list;
begin   next symbol; Arithexp; Macro (STACK);
        if last symbol = colon then begin next symbol; Arithexp;
                                          Macro (STACK)
                                    end
                               else ERRORMESSAGE (383);
        if last symbol = comma then Bound pair list
                               else if last symbol = bus
                                    then next symbol
                                    else ERRORMESSAGE (384)
end  Bound pair list;
```

```
procedure  Procedure declaration;
begin integer  n, f, count, save lnc;
      next symbol; f:= n:= Identifier;
      Skip parameter list; skip value list; skip specification list;
      if ¬ In library (n) then Mark position in name list (n);
      if in code (n)
      then Translate code
      else begin if Function (n) then Set inside declaration (n, true);
                 entrance block;
                 Macro2 (DPTR, display level);
                 Macro2 (INCRB, top of display);
                 for  count:= List length (n) step  - 1 until  1 do
                 begin f:= Next formal identifier (f);
                       if In value list (f)
                       then
                       begin if Subscrvar (f)
                             then Macro (CEN)
                             else
                             begin if Arithmetic (f)
                                   then begin if Integer (f)
                                              then Macro (CIV)
                                              else Macro (CRV)
                                        end
                                   else if Boolean (f) then Macro (CBV)
                                   else if String (f)  then Macro (CSTV)
                                                       else Macro (CLV)
                             end
                       end
                       else if Assigned to (f) then Macro (CLPN)
                                               else Macro (CEN)
                 end;
                 Macro2 (TDL, display level);
                 Macro2 (ENTRPB, local space);
                 Label list; f:= n;
                 for  count:= List length (n) step  - 1 until  1 do
                 begin f:= Next formal identifier (f);
                       if In value list (f) ∧ Subscrvar (f)
                       then begin Macro2 (TAA, f);
                                  if Integer (f) then Macro (TIAV)
                                                 else Macro (TAV)
                            end
                 end;
```

```
                    save lnc:= last lnc; last lnc:= - line counter;
                    Save and restore lnc (SLNC, n);
                    if last symbol = begin
                    then begin next symbol; if declarator last symbol
                                                then Declaration list;
                               Compound tail; next symbol
                         end
                    else Statement;
                    lnc:= last lnc:= save lnc;
                    if Function (n)
                    then begin Set inside declaration (n, false);
                               f:= Local position (n);
                               if Arithmetic (f) then Arithname (f) else
                               if Boolean (f)    then Boolname (f)
                               else begin Stringname (f); Macro (LOS) end
                         end;
                    Save and restore lnc (RLNC, n);
                    if use of counter stack  then Macro (EXITPC)
                                             else Macro (EXITP);
                    exit block
               end
end  Procedure declaration;


procedure Save and restore lnc (macro, n); integer  macro, n;
begin  if wanted ∧ Function (n) then Macro2 (macro, Local position1 (n))
end  Save and restore lnc;


procedure  Block;
begin    entrance block;
         Macro2 (TBL, display level); Macro2 (ENTRB, local space);
         Label list; Declaration list; Compound tail;
         if use of counter stack then Macro2 (EXITC, display level)
                                 else Macro2 (EXITB, display level);
         exit block
end  Block;
```

```
procedure  Declaration list;
begin    integer  future, arr dec;
         future:= arr dec:= 0;
next:    if type declarator last symbol then skip type declaration
         else
         if arr declarator last symbol
         then begin  if future ≠ 0
                     then begin Substitute (future);
                                future:= 0
                          end;
                     arr dec:= 1; Array declaration
              end
         else
         begin if future = 0 then Macro2 (JU, future);
               if last symbol = switch then Switch declaration
                                       else Procedure declaration
         end;
         if last symbol = semicolon then next symbol
                                    else ERRORMESSAGE (385);
         if declarator last symbol  then goto next;
         if future ≠ 0  then Substitute (future);
         if arr dec ≠ 0 then Macro2 (SWP, display level)
end  Declaration list;


procedure  Label list;
begin    integer  n, count;
         count:= Number of local labels;
         if count > 0
         then begin Macro2 (DECB, 2 × count);
                    Macro2 (LAD, display level);
                    n:= 0; for count:= count step - 1 until 1 do
                    begin next: n:= Next local label (n);
                                if Super local (n) then goto next;
                                if count = 1 then Macro2 (LAST, n)
                                             else Macro2 (NIL, n)
                    end
              end
end  Label list;
```

```
procedure  Program;
begin   integer  n;
        if letter last symbol
        then begin n:= Identifier;
                   if last symbol = colon
                   then Label declaration (n);
                   Program
             end
        else
        if digit last symbol
        then begin Unsigned number;
                   if in name list ∧ last symbol = colon
                   then Label declaration (integer label);
                   Program
             end
        else
        if last symbol = begin
        then begin next symbol;
                   if declarator last symbol then Block
                                             else Compound tail;
                   Macro (END)
             end
        else begin next symbol; Program end
end  Program;


procedure  Label declaration (n); integer  n;
begin   last lnc:= - line counter;
        if Subscrvar (n)       then begin ERRORMESSAGE (388);
                                          Subscripted variable (n)
                                   end
                               else Mark position in name list (n);
        if last symbol = colon then next symbol else  ERRORMESSAGE (389)
end  Label declaration;


procedure  Substitute (address); integer  address;
begin   Subst2 (Order counter, address) end  Substitute;


procedure  Subst2 (address1, address2);
value  address1, address2; integer  address1, address2;
begin   integer  instruction, instruct part, address part;
        address2:= abs (address2);
        instruction:= space[prog base + address2];
        instruct part:= instruction : d15 × d15 -
                        (if  instruction < 0 then 32767 else 0);
        address part:= instruction - instruct part;
        space[prog base + address2]:= instruct part + address1;
        if address part = 0
        then begin if instruct part = end of list
                   then space[prog base + address2]:=
                        - space[prog base + address2]
             end
        else Subst2 (address1, address part)
end  Subst2;
```

```
integer procedure  Order counter;
begin   Macro (EMPTY); Order counter:= instruct counter
end  Order counter;


procedure  Macro (macro number); integer  macro number;
begin   Macro2 (macro number, parameter) end  Macro;


procedure  Macro2 (macro number, metaparameter);
integer  macro number, metaparameter;
begin   macro:= if macro number < 512 then macro list[macro number]
                                      else macro number;
        parameter:= metaparameter;
        if state = 0
        then begin if macro = STACK then state:= 1
                   else
                   if Simple arithmetic take macro then Load (3)
                   else
                   Produce (macro, parameter)
             end
        else
        if state = 1
        then begin Load (2);
                   if ¬ Simple arithmetic take macro
                   then begin Produce (STACK, parameter); Unload end
             end
        else
        if state = 2
        then begin if Optimizable operator then Optimize
                   else
                   begin Produce (STACK, parameter); state:= 3;
                         Macro2 (macro, parameter)
                   end
             end
        else
        if state = 3
        then begin if macro = NEG then Optimize
                   else
                   begin Unload; Macro2 (macro, parameter) end
             end;
        if Forward jumping macro ∧ metaparameter ≤ 0
        then Assign (metaparameter)
end  Macro2;
```

```
procedure  Load (state i); integer  state i;
begin   stack0:= macro; stack1:= parameter; state:= state i end  Load;


procedure  Unload;
begin   Produce (stack0, stack1); state:= 0 end  Unload;


procedure  Optimize;
begin   stack0:= tabel[5 × Opt number (macro) + Opt number (stack0)];
        Unload
end  Optimize;


procedure  Assign (metaparameter); integer  metaparameter;
begin   metaparameter:= - (instruct counter - 1) end  Assign;


procedure  Produce (macro, parameter); integer  macro, parameter;
begin   integer  number, par number, entry, count;
        if macro = EMPTY then
        else
        if macro = CODE
        then begin space[prog base + instruct counter]:= parameter;
                   instruct counter:= instruct counter + 1;
                   test pointers
             end
        else begin number:= Instruct number (macro);
                   par number:= Par part (macro);
                   entry:= Instruct part (macro) - 1;
                   if par number > 0
                   then Process parameter (macro, parameter);
                   Process stack pointer (macro);
                   for count:= 1 step 1 until number do
                   Produce (CODE, instruct list[entry + count] +
                                  (if  count = par number
                                   then parameter else 0))
             end
end  Produce;
```

```
procedure  Process stack pointer (macro); integer  macro;
begin   if ⅂ in code body
        then
        begin integer  reaction;
              reaction:= B reaction (macro);
              if reaction < 9
              then begin b:= b + reaction - 4;
                         if b > max depth then max depth:= b
                   end
              else
              if reaction = 10 then b:= 0
              else
              if reaction = 11 then b:= b - 2 × (dimension - 1)
              else
              if reaction = 12
              then begin if ecount = 0
                         then
                         begin ret level:= b;
                               ret max depth:= max depth;
                               b:= 0; max depth:= max depth isr
                         end;
                         ecount:= ecount + 1
                   end
              else
              if reaction = 13
              then begin if macro = EXITSV
                         then
                         begin if b > max depth isr
                               then max depth isr:= b;
                               b:= b - 2 × (dimension - 1)
                         end;
                         if ecount = 1
                         then
                         begin if max depth > max depth isr
                               then max depth isr:= max depth;
                               b:= ret level;
                               max depth:= ret max depth
                         end;
                         if ecount > 0 then ecount:= ecount - 1
                   end
              else
              if reaction = 14
              then begin b:= display level + top of display;
                         if b > max display length
                         then max display length:= b;
                         ret max depth:= max depth
                   end
              else
              if reaction = 15
              then begin if b > max proc level
                         then max proc level:= b;
                         b:= 0; max depth:= ret max depth
                   end
        end
end  Process stack pointer;
```

```
procedure  Process parameter (macro, parameter);
integer  macro, parameter;
begin   if Value like (macro)
        then
        begin if macro = TBC
              then parameter:= if parameter = true then 0 else 1
              else
              if macro = SWP then parameter:= d9 × parameter
              else
              if macro ≠ EXITSV then parameter:= abs (parameter)
        end
        else
        begin if macro = JU ∨ macro = SUBJ ∨ macro = NIL ∨ macro = LAST
              then begin if parameter ≤ 0
                         then parameter:= - parameter
                         else parameter:= Program address (parameter)
                   end
              else parameter:= Address (parameter) +
                               (if Dynamic (parameter)
                                then (if macro = TLV ∨ macro = TAA
                                      then function digit
                                      else if macro = STST
                                           then function letter
                                           else c variant)
                                else 0)
        end
end  Process parameter;


Boolean procedure  Simple arithmetic take macro;
begin   Simple arithmetic take macro:= bit string (d1, d0, macro) = 1
end  Simple arithmetic take macro;


Boolean procedure  Optimizable operator;
begin   Optimizable operator:= bit string (d2, d1, macro) = 1
end  Optimizable operator;

Boolean procedure  Forward jumping macro;
begin   Forward jumping macro:= bit string (d3, d2, macro) = 1
end  Forward jumping macro;


Boolean procedure  Value like (macro); integer  macro;
begin   Value like:= bit string (d4, d3, macro) = 1 end  Value like;


integer procedure  Opt number (macro); integer  macro;
begin   Opt number:= bit string (d8, d4, macro) end  Opt number;


integer procedure  Instruct number (macro); integer  macro;
begin   Instruct number:= bit string (d10, d8, macro)
end  Instruct number;
```

```
integer procedure  Par part (macro); integer  macro;
begin   Par part:= bit string (d12, d10, macro) end  Par part;

integer procedure  Instruct part (macro); integer  macro;
begin   Instruct part:= bit string (d21, d12, macro) end  Instruct part;

integer procedure  B reaction (macro); integer  macro;
begin   B reaction:= macro : d21 end  B reaction;

integer procedure  Code bits (n); integer  n;
begin   Code bits:= space[nl base - n] : d19 end  Code bits;

integer procedure  Character (n); integer  n;
begin   Character:= bit string (d24, d19, space[nl base - n])
end  Character;

Boolean procedure  Arithmetic (n); integer  n;
begin   integer  i;
        i:= type bits (n);
        Arithmetic:= Character (n) ≠ 24 ∧ (i < 2 ∨ i = 4)
end  Arithmetic;

Boolean procedure  Real (n); integer  n;
begin   Real:= Character (n) ≠ 24 ∧ type bits (n) = 0 end  Real;

Boolean procedure  Integer (n); integer  n;
begin   Integer:= type bits (n) = 1 end  Integer;

Boolean procedure  Boolean (n); integer  n;
begin   Boolean:= type bits (n) = 2 end  Boolean;

Boolean procedure  String (n); integer  n;
begin   String:= type bits (n) = 3 end  String;

Boolean procedure  Designational (n); integer  n;
begin   Designational:= type bits (n) = 6 end  Designational;

Boolean procedure  Arbost (n); integer  n;
begin   Arbost:= Character (n) ≠ 24 ∧ type bits (n) < 6 end  Arbost;

Boolean procedure  Unknown (n); integer  n;
begin   Unknown:= type bits (n) = 7 end  Unknown;
```

```
Boolean procedure  Nonarithmetic (n); integer  n;
begin   integer  i;
        i:= type bits (n);
        Nonarithmetic:= Character (n) = 24 ∨ i = 2 ∨ i = 3 ∨ i = 6
end  Nonarithmetic;


Boolean procedure  Nonreal (n); integer  n;
begin   Nonreal:= Nonarithmetic (n) ∨ type bits (n) = 1 end  Nonreal;


Boolean procedure  Noninteger (n); integer  n;
begin   Noninteger:= Nonarithmetic (n) ∨ type bits (n) = 0
end  Noninteger;


Boolean procedure  Nonboolean (n); integer  n;
begin   integer  i;
        i:= type bits (n); Nonboolean:= i ≠ 2 ∧ i ≠ 5 ∧ i ≠ 7
end  Nonboolean;


Boolean procedure  Nonstring (n); integer  n;
begin   integer  i;
        i:= type bits (n); Nonstring:= i ≠ 3 ∧ i ≠ 5 ∧ i ≠ 7
end  Nonstring;


Boolean procedure  Nondesignational (n); integer  n;
begin   Nondesignational:= type bits (n) < 6 end  Nondesignational;


Boolean procedure  Nontype (n); integer  n;
begin   Nontype:= type bits (n) = 6 ∨ (Proc (n) ∧ Nonfunction (n))
end  Nontype;


Boolean procedure  Simple (n); integer  n;
begin   Simple:= Code bits (n) = 127 ∨ Simple1 (n) end  Simple;


Boolean procedure  Simple1 (n); integer  n;
begin   Simple1:= Character (n) : d3 = 0 end  Simple1;


Boolean procedure  Subscrvar (n); integer  n;
begin   Subscrvar:= Character (n) : d3 = 1 end  Subscrvar;

Boolean procedure  Proc (n); integer  n;
begin   Proc:= Character (n) : d3 > 1 ∧ Code bits (n) ≠ 127 end  Proc;


Boolean procedure  Function (n); integer  n;
begin   Function:= Character (n) : d3 = 2 end  Function;
```

```
Boolean procedure  Nonsimple (n); integer  n;
begin  Nonsimple:= ¬ (Simple (n) ∨ (if Proc (n)
                                    then (Formal (n) ∨ Function (n)) ∧
                                          List length (n) < 1
                                    else false ))
end  Nonsimple;


Boolean procedure  Nonsubscrvar (n); integer  n;
begin  Nonsubscrvar:= Simple1 (n) ∨ Proc (n) end  Nonsubscrvar;


Boolean procedure  Nonproc (n); integer  n;
begin  Nonproc:= ¬ (Character (n) : d3 > 2 ∨
                    (Formal (n) ∧ Simple1 (n) ∧ ¬ Assigned to (n)))
end  Nonproc;


Boolean procedure  Nonfunction (n);  integer  n;
begin  Nonfunction:= ¬ (Function (n) ∨ Formal (n)) end  Nonfunction;


Boolean procedure  Formal (n); integer  n;
begin  Formal:= Code bits (n) > 95 end  Formal;


Boolean procedure  In value list (n); integer  n;
begin  In value list:= Code bits (n) > 63 ∧ ¬ Formal (n)
end  In value list;


Boolean procedure  Assigned to (n); integer  n;
begin  Assigned to:= bit string (d19, d18, space[nl base - n]) = 1
end  Assigned to;


Boolean procedure  Dynamic (n); integer  n;
begin  Dynamic:= Code bits (n) > 63 ∨ Assigned to (n) end  Dynamic;


Boolean procedure  In library (n); integer  n;
begin  In library:= space[nl base - n - 1] ≥ d25 end  In library;


Boolean procedure  Id1 (k, n); integer  k, n;
begin  Id1:= bit string (2 × k, k, space[nl base - n - 1]) = 1 end  Id1;


Boolean procedure  Operator like (n); integer  n;
begin  Operator like:= Id1 (d23, n) end  Operator like;


Boolean procedure  Outside declaration (n); integer  n;
begin  Outside declaration:= Id1 (d22, n) end  Outside declaration;
```

```
Boolean procedure  Ass to function designator (n); integer  n;
begin   Ass to function designator:= Id1 (d21, n)
end Ass to function designator;


Boolean procedure  Declared (n); integer  n;
begin   Declared:= Id1 (d19, n) end  Declared;


Boolean procedure  Super local (n); integer  n;
begin   Super local:= Id1 (d18, n) end  Super local;


procedure  Change (k, n); integer  k, n;
begin   integer  i, j;
        i:= space[nl base - n - 1]; j:= i - i : (2 × k) × (2 × k);
        space[nl base - n - 1]:= i + (if  j < k then  k else  - k)
end  Change;


integer procedure  Local position (n); integer  n;
begin   if  ¬ Ass to function designator (n) then  Change (d21, n);
        Local position:= Local position1 (n)
end  Local position;


integer procedure  Local position1 (n); integer  n;
begin   Local position1:= n + 2 end  Local position1;


procedure  Set inside declaration (n, bool); integer  n; Boolean  bool;
begin   Change (d22, n);
        if  ¬ (bool ∨ Ass to function designator (n))
        then  ERRORMESSAGE (390)
end  Set inside declaration;
```

```
procedure  Mark position in name list (n); integer  n;
begin   integer  address;
        if Declared (n)
        then ERRORMESSAGE (391)
        else begin address:= Program address (n);
                   if address ≠ 0 then Substitute (address);
                   Change (d19, n)
             end
end  Mark position in name list;


integer procedure  Program address (n); integer  n;
begin   integer  word, head, m;
        m:= if Code bits (n) = 6 then n + 1 else n;
        word:= space[nl base - m]; head:= word : d18 × d18;
        if ¬ Declared (n)
        then space[nl base - m]:= head + Order counter;
        Program address:= word - head
end  Program address;


integer procedure  Address (n); integer  n;
begin   integer  word, tail, level;
        word:= Code bits (n);
        if word > 13 ∧ word < 25
        then tail:= Program address (n)
        else begin word:= space[nl base - n];
                   tail:= word - word : d18 × d18;
                   if Dynamic (n)
                   then begin level:= tail : d9;
                              if level = proc level ∧
                                 ¬ in switch declaration
                              then tail:= tail + d9 × (63 - level)
                        end
             end;
        Address:= tail
end  Address;


integer procedure  List length (n); integer  n;
begin   List length:= bit string (d18, d0, space[nl base - n - 1]) - 1
end  List length;


procedure  Test forcount (n); integer  n;
begin   if space[nl base - n - 1] : d20 > for count
        then ERRORMESSAGE (392)
end  Test forcount;
```

```
procedure  Check dimension (n); integer  n;
begin   integer  i;
        i:= if Code bits (n) = 14 then 1 else List length (n);
        if i ≥ 0 ∧ i ≠ dimension then ERRORMESSAGE (393)
end  Check dimension;


procedure  Check list length (f, n); integer  f, n;
begin   integer  i, j;
        i:= List length (f);
        j:= if Code bits (n) = 14 then 1 else List length (n);
        if i ≥ 0 ∧ j > 0 ∧ i ≠ j then ERRORMESSAGE (394)
end  Check list length;


procedure  Check type (f, n); integer  f, n;
begin   if (Designational (f) ∧ Nondesignational (n)) ∨
           (Arbost (f)        ∧ Nontype (n))          ∨
           (Arithmetic (f)    ∧ Nonarithmetic (n))    ∨
           (Boolean (f)       ∧ Nonboolean (n))       ∨
           (String (f)        ∧ Nonstring (n))
        then ERRORMESSAGE (395)
end  Check type;


integer procedure  Number of local labels;
begin   Number of local labels:=
          bit string (d13, d0, space[nl base - block cell pointer - 3])
end  Number of local labels;


integer procedure  Next local label (n); integer  n;
begin   Next local label:=
          if n = 0 then space[nl base - block cell pointer - 3] : d13
                   else next identifier (n)
end  Next local label;


integer procedure  Next formal identifier (n); integer  n;
begin   Next formal identifier:=
          next identifier (n + (if Formal (n) ∨ In library (n) ∨
                                   In value list (n)
                                then 2
                                else if Function (n) then 9 else 8))
end  Next formal identifier;


procedure  Increase status (increment); integer  increment;
begin   space[nl base - block cell pointer - 2]:=
        space[nl base - block cell pointer - 2] + increment
end  Increase status;


integer procedure  Identifier;
begin   read identifier; Identifier:= look up end  Identifier;
```

```
procedure  Skip parameter list;
begin   if last symbol = open
        then begin next symbol; skip type declaration;
                   if last symbol = close then next symbol
             end;
        if last symbol = semicolon then next symbol
end  Skip parameter list;


procedure  Translate code;
begin   integer  macro, parameter;
        if last symbol = quote
        then begin in code body:= true;
             next: next symbol;
                   if digit last symbol
                   then
                   begin macro:= unsigned integer (0);
                         if macro < 512 then macro:= macro list[macro];
                         if Par part (macro) > 0
                         then
                         begin if last symbol = comma
                               then next symbol
                               else ERRORMESSAGE (396);
                               if letter last symbol
                               then parameter:= Identifier
                               else
                               if digit last symbol
                               then parameter:= unsigned integer (0)
                               else
                               if last symbol = minus
                               then
                               begin next symbol;
                                     if digit last symbol
                                     then parameter:=
                                          - unsigned integer (0)
                                     else ERRORMESSAGE (397)
                               end
                               else ERRORMESSAGE (398);
                               Macro2 (macro, parameter)
                         end
                         else Macro (macro)
                   end
                   else ERRORMESSAGE (399);
                   if last symbol = comma then goto next;
                   if last symbol = unquote then next symbol
                                            else ERRORMESSAGE (400);
                   in code body:= false
             end
        else ERRORMESSAGE (401);
        entrance block; exit block
end  Translate code;
```

```
procedure  Unsigned number;
begin   integer  p;
        unsigned number;
        if ⌝ small
        then begin p:= 0;
             next: if p = dp0 then goto found;
                   if space[prog base + p] ≠ value of constant ∨
                      space[prog base + p + 1] ≠ decimal exponent
                   then begin p:= p + 2; goto next end;
             found: address of constant:= p
             end
end  Unsigned number;


procedure  Arithconstant;
begin   if small then Macro2 (TSIC, value of constant)
        else
        if real number then Macro2 (TRC, address of constant)
                        else Macro2 (TIC, address of constant)
end  Arithconstant;


integer procedure  Operator macro (n); integer  n;
begin   Operator macro:= space[nl base − n − 2] end  Operator macro;


procedure  Constant string;
begin   integer  word, count;
        quote counter:= 1;
next0:  word:= count:= 0;
next1:  next symbol;
        if last symbol ≠ unquote
        then begin word:= d8 × word + last symbol;
                   count:= count + 1;
                   if count = 3
                   then begin Macro2 (CODE, word); goto next0 end;
                   goto next1
             end;
next2:  word:= d8 × word + 255; count:= count + 1;
        if count < 3 then goto next2;
        Macro2 (CODE, word); quote counter:= 0; next symbol
end  Constant string;


integer procedure  Relatmacro;
begin   Relatmacro:=  if  last symbol = les then  LES else
                      if  last symbol = mst then  MST else
                      if  last symbol = mor then  MOR else
                      if  last symbol = lst then  LST else
                      if  last symbol = equ then  EQU else  UQU
end  Relatmacro;
```

```
main program of translate scan:
        if ¬ text in memory
        then begin NEWPAGE;
                   PRINTTEXT (⟨input tape for translate scan⟩)
             end;
        start:= instruct counter; last nlp:= nlp;
        runnumber:= 300; init; increment:= d13;
        state:= b:= max depth:= max depth isr:=
        max display length:= max proc level:= ecount:= 0;
        in switch declaration:= in code body:= false;
        next block cell pointer:= 0;
        entrance block; next symbol;
        Program;
        sum of maxima:= max depth + max depth isr +
                        max display length + max proc level;
        Macro2 (CODE, sum of maxima);
        output
end  translate;
```

```
procedure  output;
begin   integer  i, k, apostrophe, instruct number, par, address;

        procedure  pucar (n); integer  n;
        begin  integer  i;
               for i:= 1 step 1 until n do PUNLCR
        end  pucar;

        procedure  tabspace (n); integer  n;
        begin  integer  i, k;
               k:= n : 8;
               for i:= 1 step 1 until k do PUSYM (118);
               PUSPACE (n - k × 8)
        end  tabspace;

        procedure  absfixp (k); integer  k;
        begin  ABSFIXP (4, 0, k); pucar (2) end  absfixp;

        procedure  punch (bool); Boolean  bool;
        begin  if bool then PUTEXT (‹ true›)
                       else PUTEXT (‹false›);
               pucar (2)
        end  punch;

        procedure  punch octal (n);  value  n;  integer  n;
        begin  integer  i, k;
               Boolean  minussign;
               minussign:= n < 0; n:= abs (n);
               PUSYM (if minussign then minus else plus);
               PUSYM (apostrophe);
               for i:= d24, d21, d18, d15, d12, d9, d6, d3, d0 do
               begin k:= n : i; n:= n - k × i; PUSYM (k) end;
               PUSYM (apostrophe)
        end  punch octal;

        apostrophe:= 120;
        PUNLCR;
        if runnumber = 100
        then
        begin tabspace (22); PUTEXT (‹prescan0›); pucar (2);
              PUTEXT (‹erroneous›); PUSPACE (14);
              punch (erroneous); PUTEXT (‹text length›);
              PUSPACE (12);
              absfixp (if text in memory then text pointer + 1 else 0);
              PUTEXT (‹namelist›); pucar (2);
              for i:= 0 step 1 until nlp - 1 do
              begin tabspace (7); ABSFIXP (4, 0, i); PUSPACE (5);
                    punch octal (space[nl base - i]); PUNLCR
              end;
              STOPCODE;
              PUNLCR; PUTEXT (‹dp0›); pucar (2);
              PUTEXT (‹start›); pucar (2);
              PUTEXT (‹program›); pucar (2);
```

```
            for i:= prog base step 1 until instruct counter - 1 do
            begin tabspace (7); ABSFIXP (4, 0, i);
                  FIXP (16, 0, space[i]); PUNLCR
            end;
            RUNOUT; STOPCODE
      end
      else if runnumber = 200
      then
      begin tabspace (38); PUTEXT (‡prescan1‡); pucar (2);
            tabspace (39); punch (erroneous); tabspace (39);
            absfixp (if text in memory then text pointer + 1 else 0);
            pucar (2);
            for i:= 0 step 1 until nlp - 1 do
            begin tabspace (34); punch octal (space[nl base - i]);
                  PUNLCR
            end;
            STOPCODE; pucar (7);
            for i:= prog base step 1 until instruct counter - 1 do
            begin tabspace (32); FIXP (13, 0, space[i]); PUNLCR end;
            RUNOUT; STOPCODE
      end
      else
      begin tabspace (54); PUTEXT (‡translate‡); pucar (2);
            tabspace (55); punch (erroneous); tabspace (55);
            absfixp (if text in memory then text pointer + 1 else 0);
            pucar (2);
            for i:= 0 step 1 until nlp - 1 do
            begin tabspace (50); punch octal (space[nl base - i]);
                  PUSPACE (2); ABSFIXP (4, 0, i); PUNLCR
            end;
            STOPCODE; PUNLCR;
            tabspace (55); absfixp (dp0);
            tabspace (55); absfixp (start); pucar (2);
            for i:= prog base step 1 until start - 1 do
            begin tabspace (48); FIXP (13, 0, space[i]);
                  PUSPACE (2); ABSFIXP (4, 0, i); PUNLCR
            end;
            PUNLCR;
            for i:= start step 1 until instruct counter - 1 do
            begin k:= space[i]; par:= k : 32768;
                  address:= k - par × 32768;
                  instruct number:= par : 10;
                  par:= par - instruct number × 10;
                  tabspace (48); ABSFIXP (3, 0, instruct number);
                  ABSFIXP (1, 0, par); ABSFIXP (5, 0, address);
                  PUSPACE (2); ABSFIXP (4, 0, i); PUNLCR
            end
      end
end output;
```

```
main program:
        for n:= 0 step 1 until end of memory do space[n]:= 0;
        instruct counter:= prog base:= nlp:= 0;
        text base:= end of memory : 3;
        nl base:= end of memory;


        prescan0;
        if ¬ derroneous
        then begin  prescan1;
                    translate
             end;

endrun:
end
end
```