```
131071 -    214
MC ELAN1 ASSEMBLER AND MACRO PROCESSOR D.D. 01 10 71
    35 RGT(LU,S,+,A,PP)                    " U, S:= A, P
    40 MAC <LABEL>
                    <A + 5
                        A + 6>.
    51 SUBC(RGT(,A,-,B,))
    55 RGT(,A,-,B,)
    60 SUBC <SUBC(:LABEL)>.
    70 MACRO<'BEGIN' 'MACRO' NAAM:
                        'BEGIN'
                        A + 1
                        'END' NAAM
                        NAAM
                'END'>.
    76 NAAM
```

```
 1        1                'BEGIN' LABEL,                          " TESTPROGRAMMA MET EEN MACROPROCESSOR,
 2                                                                  " MR 011071
 3                                 'MACRO' RGT(V1,R1,S,R2,V2):
 4                                 'BEGIN' UU = '100 000', YY = '200 000', NN = '300 000',
 5                                         PP = '400 000', ZZ = '1 000 000', EE = '1 400 000'
 6                                 (<V1> + '660000400' + (1 <S> 1) * '400 000' + (:<R2> - 59) * '40' + (<V2> + :<R1> - 59))
 7                                 'END' RGT,
 8
 9                                 MACRO <PARAMETER>:                " ZAL WORDEN AANGEVRAAGD
10                                                                  " MET EEN BLOK ALS
11                                                                  " AKTUELE PARAMETER
12                                 'BEGIN'
13                                 !VON
14                                 <PARAMETER>
15                                 !VOFF
16                                 'END' MACRO,
17
18                                 MAC(PAR1,
19                                     PAR2):
20                                 'BEGIN' <PAR1>,
21
22                                         'MACRO' SUBC(PAR):      " MACRODECLARATIE BINNEN MAC
23                                         'BEGIN' NAAM
24                                 NAAM:   SUBC(MC[-1])            " DIT IS NIET DE MACRO SUBC
25                                         <PAR2>
26                                         <PAR>
27                                         'END' SUBC
28
29                                 <PAR1>: A + 2
30                                         SUBC(RGT(,A,-,B,))      " A:= -B
31                                         SUBC <SUBC(:<PAR1>)>.   " DE EERSTE SUBC IS DE MACRO,
32                                                                 " DE TWEEDE DE ELAN INSTRUKTIE
33                                 'END' MAC
34
35 *'000000':                 M[1000]: 'BEGIN' UU = '100 000', YY = '200 000', NN = '300 000',
36 *      2                           PP = '400 000', ZZ = '1 000 000', EE = '1 400 000'
37 *'001750':  '661500401'          (UU + '660000400' + (1 + 1) * '400 000' + (:A - 59) * '40' + (PP + :S - 59))
38 *      2                         'END'
39
40 *      3                         'BEGIN' LABEL,
41 *
42 *                                         'MACRO' SUBC(PAR):
43 *                                         'BEGIN' NAAM
44 *                                 NAAM:   SUBC(MC[-1])
45 *                                         A + 5
46 *                                         A + 6
47 *                                         <PAR>
48 *                                         'END' SUBC
49 *
50 *'001751':  '002000002'          LABEL: A + 2
51 *      4                                 'BEGIN' NAAM
52 *'001752':  '566475377'          NAAM:   SUBC(MC[-1])
53 *'001753':  '002000005'                  A + 5
54 *'001754':  '002000006'                  A + 6
55 *      5                                 'BEGIN' UU = '100 000', YY = '200 000', NN = '300 000',
56 *                                        PP = '400 000', ZZ = '1 000 000', EE = '1 400 000'
57 *'001755':  '660000500'          ( + '660000400' + (1 - 1) * '400 000' + (:B - 59) * '40' + ( + :A - 59))
58 *      5                         'END'
59 *      4                                 'END'
60 *      6                                 'BEGIN' NAAM
```

```
131071 -    214                                                          43

   61 *'001756':  '566475377'          NAAM:   SUBC(MC[-1])
   62 *'001757':  '002000005'                  A + 5
   63 *'001760':  '002000006'                  A + 6
   64 *'001761':  '562401751'                  SUBC(:LABEL)
   65 *       6                                'END'
   66 *
   67 *       3                        'END'
   68  '001762':  '566475377'  LABEL:  SUBC(MC[-1])                  " DIT IS NIET DE MACRO SUBC
   69
   70 *'001763':  '760060000'          !VON
   71 *       7                        'BEGIN' 'MACRO' NAAM;
   72 *                                        'BEGIN'
   73 *                                        A + 1
   74 *                                        'END' NAAM
   75 *
   76 *'001764':  '002000001'                  A + 1
   77 *       7                        'END'
   78 *'001765':  '660060000'          !VOFF
   79         1                 'END'
```

13. Programmatekst

```
integer max of namestack, max of defstack, max of actualstack, max of
pointerstack, max of savestack, stackptr, freeptr, pointerptr, saveptr,
spacecntr, lcntr, bcntr, llcntr, bbcntr, SPACEcntr, LLcntr, BBcntr, t8j,
t8J, word, Word, nextacc, endmarker, place of name, tt, asterisk, ksiretsa;
boolean in def mode, in actual mode, only mac, from macro, from
actualstack, accent read;
integer array namestack[-2:255],definitionstack[0:4095], actualstack
[0:2047], pointerstack[1:128], savestack[-1:120];

procedure initialize macro variables;
begin in def mode:= in actual mode:= only mac:= from macro:=
      from actualstack:= accent read:= false;
      max of namestack:= 255;
      max of defstack:= 4095;
      max of actualstack:= 2047;
      max of pointerstack:= 128;
      max of savestack:= 120;
      asterisk:= 254; ksiretsa:= 255;
      stackptr:= namestack[0]:= pointerstack[1]:= 0;
      endmarker:= spacecntr:= SPACEcntr:= 150;
      namestack[-1]:= saveptr:= -5;
      tt:= 1 + t6 + t12 + t18;
      freeptr:= 1
end initialize macro variables;

procedure define macro;
begin integer i,savel,max of formallist;
      boolean empty;
      integer array formallist[0:127];

      procedure read name;
      begin integer save;
            ERROR(stackptr > max of namestack,3000);
            lcntr:= savel:= namestack[stackptr]; ERROR(lcntr = -1,3027);
            bcntr:= 2; save:= stackptr:= stackptr + 1;
            store letgits(namestack,stackptr,max of namestack,reaffer);
            ERROR(stackptr + 1 > max of namestack,3000);
            namestack[stackptr]:= stackptr - save;
            namestack[stackptr + 1]:= blocknumber;
            stackptr:= stackptr + 2
      end read name;
```

```
procedure read formals;
begin integer i,ptr,aux,par;
      boolean in comma mode;

      integer procedure reaffer1;
      if symbol = accent symbol then
      begin ERROR(true,3023); goto outaccent end
      else reaffer1:= reaffer;

      procedure reaffer1 while(condition); boolean condition;
      begin integer i;
            for i:= i while condition do reaffer1
      end reaffer1 while;

      ptr:= 1; par:= 0;
      if symbol ≠ colon symbol ∧ symbol ≠ open symbol ∧ symbol ≠
      smaller symbol then
      begin ERROR(true,3026);
            reaffer1 while(symbol ≠ colon symbol ∧
            symbol ≠ open symbol ∧ symbol ≠ smaller symbol)
      end;
      if symbol ≠ colon symbol then
      begin i:= aux:= 0;
            in comma mode:= symbol = open symbol;
            for i:= i + 1 while symbol ≠ colon symbol do
            begin par= i; reaffer1;
                  if in comma mode ∧ i > 1 then reaffer1 while
                  (symbol = nlcr symbol ∨ symbol = semicolon symbol);
                  if 10 < symbol ∧ symbol ≤ 62 then
                  begin if i = 23 then
                        begin ERROR(true,3002); reaffer1 while
                              (symbol ≠ colon symbol)
                        end else
                        begin store letgits(formallist,ptr,max of
                              formallist, reaffer);
                              if ptr - aux > 22 then
                              begin ERROR(true,3011); ptr:= aux + 22
                              end;
                              formallist[aux]:= ptr - aux - 1;
                              aux:= ptr; ptr:= ptr + 1;
                              if symbol = accent symbol then
                              begin ERROR(true,3023); goto outaccent
                              end;
                              if in comma mode then
                              begin if symbol = close symbol then
                                begin if reaffer1 ≠ colon symbol then
                                  begin ERROR(true,3004);
                                    reaffer1 while
                                    (symbol ≠ colon symbol)
                                  end
                                end else if symbol ≠ comma symbol then
                                begin ERROR(true,3005);
                                  reaffer1 while(symbol ≠ comma
                                  symbol ∧ symbol ≠ colon symbol)
                                end
                              end else
```

```
                                begin if symbol ≠ greater symbol then
                                  begin ERROR(true,3006);
                                    reafferl while(symbol ≠ smaller
                                    symbol ∧ symbol ≠ colon symbol)
                                  end else
                                  begin if reafferl ≠ colon symbol then
                                    begin reafferl while(symbol =
                                      nlcr symbol ∨ symbol = semicolon
                                      symbol);
                                      ERROR(symbol = colon symbol,3007)
                                    end;
                                    if symbol ≠ smaller symbol ∧
                                    symbol ≠ colon symbol then
                                    begin ERROR(true,3007);
                                      reafferl while(symbol ≠
                                      smaller symbol ∧ symbol ≠
                                      colon symbol)
                                    end
                                  end
                                end
                          end
                    end else
                    begin ERROR(true,3008);
                          reafferl while(symbol ≠ comma symbol ∧
                          symbol ≠ smaller symbol ∧ symbol ≠ colon
                          symbol)
                    end
              end
        end if symbol;
        if ptr - 1 < max of formallist then formallist[ptr - 1]:= 0
        else ERROR(true,3009);
        reaffer;
        ERROR(symbol ≠ nlcr symbol ∧ symbol ≠ semicolon symbol,3024);
        read while(symbol = nlcr symbol ∨ symbol = semicolon symbol);
  outaccent: definitionstack[lcntr]:= -par - 1
end read formals;

procedure read block;
begin integer i, begcntr;
      boolean declarations,within accents;
      procedure compare parameters;
      begin integer i, j, l, ptr, length;
            boolean found;
            integer array parameter[0:20];
```

```
        ptr:= l:= 0; j:= 127; found:= false;
        reaffer;
        store letgits(parameter,l,20,reaffer);
        if l ≠ 22 ∧ symbol = greater symbol then
        for length:= formallist[ptr] while length ≠ 0 ∧ ¬ found do
        begin j:= j + 1;
              if length = l then
              begin i:= 0;
                    for i:= i + 1 while parameter[i - 1] =
                    formallist[ptr + i] ∧ ¬ found do
                    if i = l then
                    begin for l:= 1, l while l ≠ smaller symbol
                          do delete symbol(l); i:= i - 1;
                          stow into stack(definitionstack,
                          max of defstack,j);
                          found:= true
                    end
              end;
              ptr:= ptr + length + 1
        end
end compare parameters;

procedure delete symbol(s); integer s;
begin integer word;
      if bcntr = 0 then
      begin s:= definitionstack[lcntr];
            lcntr:= lcntr - 1; bcntr:= 2
      end else
      begin word:= definitionstack[lcntr];
            if word < 0 then empty:= true else
            begin definitionstack[lcntr]:= word : t8;
                  s:= word - definitionstack[lcntr] × t8;
                  bcntr:= bcntr - 1
            end
      end
end delete symbol;

in def mode:= true;
stow into stack(definitionstack,max of defstack,symbol);
if ¬ compare(≮'begin'≯) then
begin ERROR(true,3010); skip until(≮'begin'≯) end;
begcntr:= 1;
declarations:= symbol ≠ nlcr symbol ∧ symbol ≠ semicolon
symbol; if ¬ declarations then
begin lcntr:= savel; bcntr:= 2;
      in def mode:= false;
      reaffer;
      stow into stack(definitionstack,max of defstack,
      symbol); in def mode:= true
end;
```

```
            within accents:= false;
            for i:= i while begcntr > 0 do
            begin read while(symbol ≠ accent symbol
                        ∧ symbol ≠ smaller symbol);
                  if symbol = smaller symbol then compare parameters;
                  if symbol = accent symbol then
                  begin within accents:= ¬ within accents;
                        reaffer;
                        if within accents then
                        begin if compare(‹end›) then
                              begin if symbol = accent symbol then
                                    begin begcntr:= begcntr - 1;
                                      if begcntr = 0 ∧ ¬ declarations
                                      then
                                      begin delete symbol(i);
                                        for i:= i while i ≠ nlcr symbol
                                        ∧ i ≠ semicolon symbol ∧ ¬ empty
                                        do delete symbol(i);
                                        empty:= false
                                      end
                                    end
                              end else
                              if compare(‹begin›) then
                              begin if symbol = accent symbol then
                                    begcntr:= begcntr + 1
                              end
                        end
                  end
            end;
            in def mode:= false; reaffer;
            stow into stack(definitionstack,max of defstack,endmarker);
            if stackptr < max of namestack then
            namestack[stackptr]:= if lcntr + 1 > max of defstack
                                  then -1 else lcntr + 1
      end read block;
      max of formallist:= 127; empty:= false;
      for i:= i while ¬ empty do
      begin read name;
            read formals;
            read block;
            read while(0 ≤ symbol ∧ symbol ≤ 62);
            if symbol = comma symbol then
            begin reaffer;
                  read while(symbol = nlcr symbol
                              ∨ symbol = semicolon symbol);
                  ERROR(symbol < 10 ∨ symbol > 62,3030)
            end else empty:= true
      end;
      pr tape symbol:= space symbol
end define macro;
```

```
procedure expand macro;
begin integer p,par;

      procedure read actuals;
      begin integer i,opcntr,quotcntr,savel,auxptr;

            procedure complete actual parameter;
            begin if bcntr = 0 then
                  begin lcntr:= lcntr - 1; bcntr:= 2 end else
                  begin actualstack[lcntr]:= actualstack[lcntr] : t8;
                        bcntr:= bcntr - 1
                  end;
                  stow into stack(actualstack,max of actualstack,
                  endmarker);
                  freeptr:= freeptr + 1;
                  if freeptr < max of pointerstack then
                  begin savel:= lcntr;
                        pointerstack[freeptr]:= lcntr + 1
                  end
            end complete actual parameter;

            auxptr:= freeptr;
            if symbol = open symbol then
            begin in actual mode:= true;
                  opcntr:= 1;
                  for i:= i while opcntr > 0 do
                  begin ERROR(freeptr > max of pointerstack,3013);
                        if freeptr = auxptr then
                        begin savel:= lcntr:= pointerstack[freeptr] - 1;
                              bcntr:= 2
                        end;
                        reaffer;
                        if symbol = open symbol then
                        opcntr:= opcntr + 1 else
                        if symbol = close symbol then
                        opcntr:= opcntr - 1;
                        read while(symbol = nlcr symbol ∨
                                   symbol = semicolon symbol);
                        lcntr:= savel; bcntr:= 2;
                        stow into stack(actualstack,max of actualstack,
                        symbol);
                        for i:= i while (symbol ≠ comma symbol ∨ opcntr
                        ≠ 1) ∧ opcntr ≠ 0 do
                        begin reaffer;
                              if symbol = open symbol then opcntr:=
                                opcntr + 1 else
                              if symbol = close symbol then opcntr:=
                                opcntr - 1
                        end;
                        complete actual parameter
                  end;
                  reaffer;
                  in actual mode:= false
            end else
```

```
      if symbol = smaller symbol then
      begin in actual mode:= true;
            for i:=i while symbol = smaller symbol do
            begin ERROR(freeptr > max of pointerstack,3013);
                  quotcntr:= 1;
                  if freeptr = auxptr then
                  lcntr:= pointerstack[freeptr] - 1 else lcntr:= savel;
                  bcntr:= 2;
                  for i:= i while quotcntr > 0 do
                  begin reaffer;
                        if symbol = smaller symbol then quotcntr:=
                          quotcntr + 1 else
                        if symbol = greater symbol then quotcntr:=
                          quotcntr - 1
                  end;
                  complete actual parameter;
                  reaffer;
                  if symbol ≠ point symbol then
                  begin read while(symbol = nlcr symbol
                                   ∨symbol = semicolon symbol);
                        ERROR(symbol = point symbol,3025)
                  end
            end;
            if symbol = point symbol then reaffer
            else ERROR(true,3025);
            in actual mode:= false
      end;
      pointerptr:= auxptr;
      if freeptr - auxptr ≠ par then
      begin ERROR(true,3016);
            auxptr:= auxptr + par - 1;
            for i:= freeptr step 1 until auxptr do
            pointerstack[i]:= -1;
            freeptr:= auxptr + 1
      end;
      if symbol ≠ nlcr symbol ∧ symbol ≠ semicolon symbol then
      begin ERROR(true,3001);
            read while(symbol ≠ nlcr symbol
                    ∧ symbol ≠ semicolon symbol)
      end
end read actuals;

procedure store expansion;
begin savestack[saveptr]:= bbcntr;
      savestack[saveptr + 1]:= llcntr;
      if from actualstack then
      begin savestack[saveptr + 2]:= BBcntr;
            savestack[saveptr + 3]:= LLcntr;
            from actualstack:= false
      end else savestack[saveptr + 2]:= -1
end store expansion;
```

```
      ERROR(saveptr + 5 > max of actualstack,3017);
      p:= namestack[place of name - namestack[place of name] - 1];
      par:= -definitionstack[p] - 1;
      read actuals;
      namestack[place of name + 1]:= -namestack[place of name + 1];
      savestack[saveptr + 4]:= place of name;
      savestack[saveptr + 5]:= symbol;
      if from macro then store expansion else
      begin from macro:= true; stow into buffer(asterisk) end;
      saveptr:= saveptr + 6; bbcntr:= 1; llcntr:= p;
      pr tape symbol:= space symbol;
      symbol:= macro sym;
      stow into buffer(symbol)
end expand macro;

integer procedure macro sym;
begin integer i,s;

      procedure restore expansion;
      begin bbcntr:= savestack[saveptr];
            llcntr:= savestack[saveptr + 1];
            if bbcntr = 2 then
            begin t8j:= 1; word:= definitionstack[llcntr];
                  word:= word - word : t8 × t8
            end else
            if bbcntr = 3 then
            begin t8j:= t8; word:= definitionstack[llcntr];
                  word:= word - word : t16 × t16
            end;
            if savestack[saveptr + 2] ≠ -1 then
            begin BBcntr:= savestack[saveptr + 2];
                  LLcntr:= savestack[saveptr + 3];
                  if BBcntr = 2 then
                  begin t8J:= 1; Word:= actualstack[LLcntr];
                        Word:= Word - Word : t8 × t8
                  end else
                  if BBcntr = 3 then
                  begin t8J:= t8; Word:= actualstack[LLcntr];
                        Word:= Word - Word : t16 × t16
                  end;
                  from actualstack:= true
            end;
            place of name:= savestack[saveptr - 2];
            pointerptr:= pointerptr + definitionstack[namestack
            [place of name - namestack[place of name] - 1]] + 1
      end restore expansion;
```

```
      if spacecntr > 150 then
      begin spacecntr:= spacecntr - 1; s:= space symbol end else
      begin if from actualstack then
            begin BBcntr:= BBcntr - 1; if BBcntr = 0 then
                  begin LLcntr:= LLcntr + 1; BBcntr:= 3;
                        t8J:= t16; Word:= actualstack[LLcntr]
                  end;
                  if BBcntr ≠ 1 then
                  begin s:= Word : t8J; Word:= Word - s × t8J;
                        t8J:= t8J / t8
                  end else s:= Word;
                  if s = endmarker then
                  begin from actualstack:= false; s:= macro sym
                  end
            end else
            begin bbcntr:= bbcntr - 1; if bbcntr = 0 then
                  begin llcntr:= llcntr + 1; bbcntr:= 3; t8j:= t16;
                        word:= definitionstack[llcntr]
                  end;
                  if bbcntr ≠ 1 then
                  begin s:= word : t8j; word:= word - s × t8j;
                        t8j:= t8j / t8
                  end else s:= word;
                  if s > 128 ∧ s < 149 then
                  begin from actualstack:= true; BBcntr:= 1;
                        LLcntr:= pointerstack[pointerptr + s - 128] - 1;
                        if LLcntr = -2 then from actualstack:= false;
                        s:= macro sym
                  end else
                  if s = endmarker then
                  begin saveptr:= saveptr - 6;
                        freeptr:= pointerptr;
                        namestack[place of name + 1]:=
                        -namestack[place of name + 1];
                        if saveptr = -5 then
                        begin from macro:= false; stow into buffer(ksiretsa)
                        end else restore expansion;
                        s:= savestack[saveptr + 5]
                  end
            end;
            if s > endmarker then
            begin spacecntr:= s - 1; s:= space symbol end
      end;
      macro sym:= s
end macro sym;
```

```
integer procedure reaffer;
begin integer i;

      integer procedure read and buffer;
      begin integer s;
            s:= RESYM1;
            if in actual mode then
            begin stow into stack(actualstack,max of actualstack,s);
                  prsym(s);
                  if s = nlcr symbol then space(7)
            end else
            begin stow into buffer(s);
                  if s = nlcr symbol then
                  line number:= line number + 1
            end;
            read and buffer:= s
      end read and buffer;

      for i:= i,i while symbol = space symbol ∨ symbol = tab symbol do
      begin if accent read then
            begin symbol:= nextacc; accent read:= false end
            else symbol:= read and buffer;
            if symbol = accent symbol then
            begin nextacc:= read and buffer;
                  if nextacc = accent symbol
                  then symbol:= apostrophe symbol
                  else accent read:= true
            end;
            if symbol = apostrophe symbol then
            for i:= i while symbol ≠ semicolon symbol
                        ∧ symbol ≠ nlcr symbol do
            symbol:= read and buffer;
            if in def mode then
            stow into stack(definitionstack,max of defstack,symbol)
      end;
      reaffer:= symbol
end reaffer;

boolean procedure compare(text); string text;
begin integer s,k;
      k:= 0; compare:= true;
      for s:= stringsymbol(k,text) while s ≠ 255 do
      if s ≠ (if 37 ≤ symbol ∧ symbol ≤ 62 then symbol - 27 else symbol)
      then
      begin compare:= false; k:= -1 end else
      begin k:= k + 1; if first scan then reaffer else NS end
end compare;

procedure read while(condition); boolean condition;
begin integer i;
      for i:= i while condition do if first scan then reaffer else NS
end read while;
```

```
procedure skip until(text); string text;
begin integer i, first symbol;
      first symbol:= stringsymbol(0,text);
      read while(first symbol ≠
          (if 37 < symbol ∧ symbol < 62 then symbol - 27 else symbol));
      for i:= i while ¬ compare(text) do read while(first symbol ≠
          (if 37 < symbol ∧ symbol < 62 then symbol - 27 else symbol))
end skip until;

procedure stow into stack(stack,max,char); value max,char;
integer max,char; integer array stack;
begin integer i;
      if char = space symbol ∧ spacecntr < 255
      then SPACEcntr:= SPACEcntr + 1 else
      begin bcntr:= bcntr + 1;
            if bcntr = 3 then
            begin lcntr:= lcntr + 1; bcntr:= 0;
                  if lcntr > max then ERROR(true,3018)
                  else stack[lcntr]:= 0
            end;
            if SPACEcntr > 150 then
            begin stack[lcntr]:= stack[lcntr] × t8 + SPACEcntr;
                  if char = space symbol then SPACEcntr:= 151 else
                  begin SPACEcntr:= 150;
                        stow into stack(stack,max,char)
                  end
            end else
            stack[lcntr]:= stack[lcntr] × t8 + char;
            if char = endmarker then
            for i:= bcntr step 1 until 1 do
            stow into stack(stack,max,0)
      end
end stow into stack;

procedure store letgits(list,pointer,max,letgit); value max;
integer pointer,max,letgit; integer array list;
begin integer i,j,word;
      boolean full;
      word:= j:= 0; full:= false;
      for i:= i while symbol < 62 ∧ ¬ full do
      begin if symbol > 37 then symbol:= symbol - 27;
            j:= j + 1;
            if j = 4 then
            begin if pointer > max then full:= true else
                  list[pointer]:= word × t6 + symbol;
                  word:= j:= 0; pointer:= pointer + 1
            end else word:= word × t6 + symbol;
            symbol:= letgit
      end;
      if j ≠ 0 then
      begin for j:= j + 1 while j < 4 do word:= word × t6 + 63;
            if pointer > max then full:= true else list[pointer]:= word;
            pointer:= pointer + 1
      end;
      ERROR(full ∧ ¬ in def mode,3019)
end store letgits;
```

```
procedure unstack macros;
begin integer i;
      for i:= i while abs(namestack[stackptr - 1]) = blocknumber ∧
        stackptr > 0 do
      stackptr:= stackptr - namestack[stackptr - 2] - 3
end unstack macros;

procedure skip macro declarations;
if second scan then
begin integer i, begcntr;
      for i:= i,i while symbol = comma symbol do
      begin skip until(≮'begin'≯); begcntr:= 1;
            for i:= i while begcntr > 0 do
            begin read while(symbol ≠ accent symbol
                        ∧ symbol ≠ apostrophe symbol);
                  if symbol = accent symbol then
                  begin NS;
                        if symbol = accent symbol
                        then symbol:= apostrophe symbol else
                        if compare(≮end'≯) then begcntr:= begcntr - 1 else
                        if compare(≮begin'≯) then begcntr:= begcntr + 1
                        else begin read while(symbol ≠ accent symbol);
                                   NS
                        end
                  end;
                  if symbol = apostrophe symbol then
                  read while(symbol ≠ nlcr symbol
                           ∧ symbol ≠ semicolon symbol)
            end;
            read while(0 ≤ symbol ∧ symbol ≤ 62)
      end
end skip macro declarations;

procedure print elantext;
begin integer i, begcntr;
      pr tape symbol:= space symbol;
      linecounter:= 0;
      skip until(≮'begin'≯); begcntr:= 1;
      for i:= i while begcntr > 0 do
      begin read while(symbol ≠ accent symbol
                  ∧ symbol ≠ apostrophe symbol);
            if symbol = accent symbol then
            begin NS;
                  if symbol = accent symbol
                  then symbol:= apostrophe symbol else
                  if compare(≮end'≯) then begcntr:= begcntr - 1 else
                  if compare(≮begin'≯) then begcntr:= begcntr + 1 else
                  begin read while(symbol ≠ accent symbol); NS end
            end;
            if symbol = apostrophe symbol then
            read while(symbol ≠ nlcr symbol ∧ symbol ≠ semicolon symbol)
      end;
      runout; runout
end print elantext;
```