STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM
REKENAFDELING

MR 98

ALGOL editor

A program for standardizing program lay-out

by

H.L. Oudshoorn,
H.N. Glorie and G.C.J.M. Nogarede

RA

September 1968.

The Mathematical Centre at Amsterdam, founded the 11th of February, 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for Pure Research (Z.W.O.) and the Central National Council for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

Acknowledgement

The authors are indebted to Prof. Dr. F.E.J. Kruseman Aretz and Dr. T.J. Dekker for their invaluable suggestions and criticisms, to P. Beertema for his help in testing the program, to Drs. W.J. Laan for carefully reading the manuscript, to Miss J. Koerts for typing the manuscript, and to D. Zwarst and J. Suiker for the printing and binding of this report.

Contents

1. Introduction and users' description.

The purpose of the program ALGOL editor is to provide, under strict rules, the text of an arbitrary number of programs, written in ALGOL 60, with lay-out symbols (i.e. spaces and carriage returns).

The authors have tried to lay down these rules in such a way that the structure of the edited program is brought out as clearly as possible. To the user a number of liberties are left, which will be discussed in the following section.

The idea of writing an ALGOL-editing program was born mainly to avoid the very scrupulous and time wasting work of preparing a program for publication. It may also be used to get a better insight into the structure of a very compactly written ALGOL-text.

First we tried to rewrite the procedure 'Algoledit' by W.M.McKeeman [1] in a form usable in the MC-ALGOL 60 system for the EL X8, but because of the rather limited possibilities of 'Algoledit' we turned over to a completely different program which is given in this report.

The data tape must contain the following control numbers:

1. numofpr : the number of programs to be edited.
2. width : the maximal number of positions on a line.
3. page : the number of lines per typing area.
4. rest : the number of blank lines between two pages.
5. n : the upper index of an integer array pos (see 6.)
6. pos [1:n] : n numbers with which the array elements pos [1] through pos [n] are filled.

The data 1 - 4 speak for themselves.
To explain the use of the array pos [1:n], which regulates the depth of the indentations we first give the following definition:

To each block or compound statement corresponds a number, called its *level*, which is defined by:

(i) for the program the level is zero;

(ii) for each other block or compound statement the level is one more than the level of the smallest embracing block or compound statement.

The depth of the indentation belonging to the text between a begin - end pair with level = k, equals pos [k], so that the first non-layout symbol of a line will be at position

$$\sum_{i=0}^{k} \text{pos } [i] \ ,$$

in which formula, by definition, pos [0] = 0 and pos [i] = pos [n] for $i \geq n$. The only exception to this rule is a label between a begin - end pair with level = $k \geq 1$; the first symbol of this label will be at position

$$\sum_{i=0}^{k-1} \text{pos } [i].$$

For example, the data tape used for the text of Algoleditor as published on pp. 7 - 16 of this report reads:

numofpr 1
width 70
page 55
rest 11
n 3
pos [1:n] 2, 4, 3.

Input. ALGOL editor requires as input:
(i) a data tape as described above,
(ii) the program(s) to be edited,
both on a tape punched in MC-flexowriter code.

Output. The output consists of a tape on which the edited programs are punched in MC-flexowriter code. At the same time, the text of the output-tape is printed over the lineprinter.

N.B. The user has to take care that the elements of the array pos are chosen in such a way that the difference

$$\text{width} - \sum_{i=0}^{\text{maximal level}} \text{pos}\,[i]$$

does not become too small, in order to retain enough space for the text of the inner block or compound statement.

If the difference becomes too small, ALGOL editor stops editing and prints:

the chosen indentations are too large with respect to
the maximal number of positions on the line.

2. Standard lay-out produced by ALGOL editor.

The way in which a program is processed by ALGOL editor is very straightforward. No deep investigations into the structure of expressions etc. are made, and decisions concerning the insertion of lay-out symbols are often taken with regard to two consecutive basic symbols only.

ALGOL editor skips all lay-out symbols except in the following cases:

(i) within an identifier and comment, one or more consecutive lay-out symbols are replaced by one space;

(ii) the text of a string remains unchanged.

The symbols are added one after another to the line under construction. As soon as the maximal width is exceeded, a new line carriage return is inserted after the semicolon added most recently, if any, or, otherwise, after the space inserted most recently.

ALGOL editor inserts a new line carriage return

before: 1) a label,
2) 'begin', unless preceded by a short label,
3) 'for', unless preceded by a label or 'begin',
4) 'end', unless it can be placed on the same line as the corresponding 'begin',
5) 'else', when preceded by 'end',
6) declarations, unless preceded by 'begin', and

after: 1) comment,
2) 'end;'.

ALGOL editor may insert a space:

1) an operator is separated from an identifier, an unsigned number, a logical value or a second operator by one space (exception: after the ¬ operator no space is inserted);
2) a declarator or specificator is followed by one space;
3) the separators step, until and while behave like the operators in 1);
4) between subscript-brackets the separators colon and comma are neither preceded nor followed by a space; in all other cases the colon and comma, just like the semicolon and := are followed by one space;
5) brackets are neither preceded nor followed by a space, except 'begin' and 'end', viz.
   a) 'begin' is followed by at least one space,
   b) 'end' is preceded by one space if its position is not at the beginning of the line;
6) a new line carriage return is followed directly by the number of tabulations and spaces of the margin.

ALGOL editor inserts a RUNOUT

1) before and after every procedure,
2) between two pages.

3. Some features of the MC-ALGOL 60 system for the Electrologica X8.

3.1. In the MC-ALGOL 60 system for the EL X8 computer [2] some procedures written in machinecode are available without declarations.
From these we use the input-procedures RESYM and READ, the output-procedures PRSYM(n), PUSYM(n), SPACE(n), PUSPACE(n), PRINTTEXT(s), NLCR, PUNLCR, RUNOUT, STOPCODE and the procedure EXIT.
The effect of a call of one of these procedures is described in Appendix I.

3.2. The hardware representations of some basic symbols in the input-tape.

The symbols :=, ÷ , ≡ , ⊃ , ≠ , ≤ , ≥ , ‘ and ’ are punched in two punchings:

:=	as	:	followed	by	= ,
÷	"	_	"	:	,
≡	"	_	"	=	,
⊃	"	_	"	¬	,
≠	"	|	"	=	,
≤	"	_	"	<	,
≥	"	_	"	>	,
‘	"	|	"	<	,
’	"	|	"	>	.

```
begin comment ALGOL editor. The data tape must contain the
  following control numbers: 1. numofpr: the number of programs to
  be edited, 2. width: the maximal number of positions on a line,
  3. page: the number of lines per type area, 4. rest: the number
  of blank lines between two pages, 5. n: the upper index of the
  integer array pos (see 6.), 6. pos[1:n]: n numbers which
  regulate the indentations;
  integer symbol, i, s, ih, sh, breaki, breaks, tabstop, level,
  arlevel, stringlevel, line, comm, decl, lab, proc, pointer, a, b,
  c, h, k, w, zz, nop, numofpr, width, page, rest, n;
  boolean booll, boolb, boolc;
  integer array buffer[1:2000], stock[1:40], proclevel[0:10];
  numofpr:= read; width:= read; page:= read; rest:= read; n:= read;
  begin integer array pos[0:n];

      procedure sym(n); value n; integer n;
      begin PUSYM(n); PRSYM(n) end;


      procedure space(n); value n; integer n;
      begin integer i;
         for i:= 1 step 1 until n do sym(93)
      end;


      procedure tabspace(n); value n; integer n;
      begin integer p, q;
         p:= n : 8; q:= n - p × 8;
         for a:= 1 step 1 until p do sym(118); space(q)
      end;


      procedure punchline(border); value border; integer border;
      begin tabspace(tabstop - (if boolb ∧ booll then (pos[b] + (if
         level ≤ n then pos[b - 1] else pos[n])) else if booll ∨
         boolb then pos[b] else 0)); booll:= boolb:= false;
         for a:= 1 step 1 until border do sym(buffer[a]); newline
      end;


      procedure newline;
      begin sym(119); line:= line + 1; if line ≥ page then
         begin RUNOUT;
            for a:= 1 step 1 until rest do sym(119); line:= 0
         end
      end;


      procedure punchbuffer;
      begin punchline(i - 1); i:= 1; s:= tabstop; breaki:= breaks:= 0
      end;


      procedure restbuffer(n); value n; integer n;
```

```
begin i:= i - n - 1;
   for a:= 1 step 1 until i do buffer[a]:= buffer[n + a];
   i:= i + 1
end;


procedure label;
begin if i < pos[b] then
   begin for a:= i step 1 until pos[b] do buffer[a]:= 93;
      i:= pos[b] + 1
   end
   else
   begin buffer[i]:= 93; i:= i + 1 end;
   lab:= 0
end;


procedure breakbuffer;
begin integer n;
   if breaki > 0 ∧ zz = 0 then
   begin punchline(breaki); restbuffer(breaki);
      s:= s - breaks + tabstop; if lab = 1 then
      begin label; s:= i + tabstop end
   end
   else
   begin for a:= i - 2 step - 1 until 1 do if buffer[a] = 93
      then
      begin n:= a; goto break1 end;
      goto break2;
   break1: punchline(n); restbuffer(n);
   break2: if lab = 1 then label; s:= i + tabstop;
      if s ≥ width then emergency
   end;
   breaki:= breaks:= 0
end;


procedure stockbuffer(spacesbefore, spacesafter, c);
value spacesbefore, spacesafter, c;
integer spacesbefore, spacesafter, c;
begin for a:= 1 step 1 until spacesbefore do buffer[i + a -
   1]:= 93; i:= i + spacesbefore; s:= s + spacesbefore;
   for a:= 1 step 1 until c do buffer[i + a - 1]:= stock[a];
   i:= i + c; s:= s + c ; 2;
   for a:= 1 step 1 until spacesafter do buffer[i + a - 1]:= 93;
   i:= i + spacesafter; s:= s + spacesafter;
   if comm = 0 then goto start1
end;


procedure semicolon;
begin buffer[i]:= 91; i:= i + 1; zz:= 0; punchbuffer;
   if level = proclevel[pointer] then
   begin RUNOUT; newline; pointer:= pointer - 1;
      proc:= if pointer = 0 then - 1 else 1
```

```
        end;
        goto start
      end;


      integer procedure undsym;
      begin
      undl: symbol:= RESYM; if symbol = 126 then goto undl;
        undsym:= symbol
      end;


      procedure emergency;
      begin ih:= i; sh:= s;
        for a:= ih step - 1 until 1 do
        begin ih:= ih - 1; sh:= sh - 1;
          if buffer[a] = 87 ∧ sh < width then
          begin punchline(ih + 1); restbuffer(ih + 1);
            s:= i + tabstop; breaki:= breaks:= 0; goto endem
          end;
          k:= a
        end;
        if k = 1 then
        begin sym(119); PRINTTEXT(
*
the chosen indentations are too large with respect to
the maximal number of positions on the line*
          ); EXIT
        end;
      endem:
      end;

  begin of program: for a:= 1 step 1 until n do pos[a]:= read;
      nop:= 0;
  repeat: i:= s:= line:= 1; booll:= boolb:= boolc:= false;
      breaki:= breaks:= tabstop:= level:= arlevel:= stringlevel:=
      comm:= decl:= lab:= b:= pointer:= zz:= proclevel[0]:=
      pos[0]:= 0; proc:= - 1; RUNOUT; sym(119);
  start: symbol:= RESYM;
      if i > 1 ∧ (symbol = 93 ∨ symbol = 119) then
      begin if buffer[i - 1] > 63 then goto start else
        begin for symbol:= RESYM while symbol = 93 ∨ symbol = 119
          do ; if symbol < 63 then
          begin buffer[i]:= 93; buffer[i + 1]:= symbol; i:= i + 2;
            s:= s + 2; goto start
          end
        end
      end;
  start1: if symbol = 93 ∨ symbol = 118 ∨ symbol = 119 then goto
      start; if s ≥ width then
      begin breakbuffer; goto start1 end;
      if symbol < 64 ∨ symbol = 76 ∨ symbol = 88 ∨ symbol = 89 ∨
      symbol = 98 ∨ symbol = 99 then
      begin comment digits, letters, ¬, ., ₁₀, (, );
        buffer[i]:= symbol; i:= i + 1; s:= s + 1; goto start
```

```
end
else if symbol = 64 ∨ symbol = 65 ∨ symbol = 66 ∨ symbol = 67
∨ symbol = 70 ∨ symbol = 72 ∨ symbol = 74 ∨ symbol = 79 ∨
symbol = 80 then
begin comment +, −, ×, /, =, <, >, ∧, ∨;
   if buffer[i − 1] ≠ 93 then
   begin buffer[i]:= 93; i:= i + 1; s:= s + 1 end;
   buffer[i]:= symbol; buffer[i + 1]:= 93; i:= i + 2; s:= s + 2;
   goto start
end
else if symbol = 100 ∨ symbol = 101 then
begin comment [, ];
   buffer[i]:= symbol; i:= i + 1; s:= s + 1;
   arlevel:= (if symbol = 100 then 1 else − 1) + arlevel;
   goto start
end
else if symbol = 87 then
begin comment ,;
   buffer[i]:= symbol; if arlevel = 0 then
   begin buffer[i + 1]:= 93; i:= i + 2; s:= s + 2 end
   else
   begin i:= i + 1; s:= s + 1 end;
   goto start
end
else if symbol = 91 then
begin comment semicolon;
   buffer[i]:= symbol; buffer[i + 1]:= 93; if s < width then
   begin breaki:= i + 1; breaks:= s + 1 end;
   i:= i + 2; s:= s + 2; if decl = 1 then
   begin if proc ≠ 0 then punchbuffer; decl:= 0; goto start
   end;
   if proc = 0 then
   begin punchbuffer; RUNOUT; newline;
      proc:= if pointer = 0 then − 1 else 1
   end;
   goto start
end
else if symbol = 90 then
begin comment colon;
   if arlevel ≠ 0 then
   begin buffer[i]:= 90; i:= i + 1; s:= s + 1; goto start end
   else
   begin symbol:= RESYM; if symbol = 70 then
      begin buffer[i]:= 90; buffer[i + 1]:= 70;
         buffer[i + 2]:= 93; i:= i + 3; s:= s + 3; goto start
      end
      else
   pardel: if symbol = 93 then
      begin symbol:= RESYM; goto pardel end;
      if symbol = 98 then
      begin buffer[i]:= 90; buffer[i + 1]:= 93;
         buffer[i + 2]:= 98; i:= i + 3; s:= s + 3; goto start
      end
      else
      begin buffer[i]:= 90; i:= i + 1; if level = 0 then
```

```
            begin punchbuffer; goto start1 end;
            lab:= 1;
            for a:= i - 2 step - 1 until 1 do if buffer[a] = 126
            ∨ buffer[a] = 91 ∨ buffer[a] = 90 then goto labinst;
            label; booll:= true; goto start1;
         labinst: if buffer[a] = 126 then breaki:= a + 2 else
            if buffer[a] = 90 then
            begin if a < pos[b] then breaki:= pos[b] else
               breaki:= a + 1
            end;
            breakbuffer; booll:= true; goto start1
         end
      end
   end
else if symbol = 127 then
begin comment |;
bar: symbol:= RESYM; if symbol = 127 then goto bar;
   if symbol = 80 ∨ symbol = 70 then
   begin comment ∧, ≠ ;
      buffer[i]:= 93; buffer[i + 1]:= 127;
      buffer[i + 2]:= symbol; buffer[i + 3]:= 93; i:= i + 4;
      s:= s + 3; goto start
   end
   else if symbol = 72 then
   begin comment ≤;
      buffer[i]:= 127; buffer[i + 1]:= 72; breaki:= i - 1;
      breaks:= s - 1; i:= i + 2; s:= s + 1;
   string: symbol:= RESYM; buffer[i]:= symbol;
      if symbol ≠ 127 then
      begin i:= i + 1; if symbol ≠ 126 then
         begin if symbol = 118 then s:= s + 8 else s:= s + 1;
            if symbol = 119 then line:= line + 1
         end;
         if s ≥ width - 4 ∧ breaks > 0 then breakbuffer;
         goto string
      end
      else
      begin
      bar1: symbol:= RESYM; if symbol = 127 then goto bar1;
         if symbol = 74 then
         begin comment ≥;
            buffer[i + 1]:= symbol; i:= i + 2; s:= s + 1;
            if stringlevel ≠ 0 then
            begin stringlevel:= stringlevel - 1; goto string
            end
            else if s > width - 3 then
            begin for a:= 1 step 1 until i - 1 do
               sym(buffer[a]); newline; i:= 1; s:= tabstop
            end;
            goto start
         end
         else if symbol = 72 then
         begin comment nested stringquotes;
            stringlevel:= stringlevel + 1;
            buffer[i + 1]:= symbol; i:= i + 2; s:= s + 1;
```

```
            goto string
         end
         else
         begin buffer[i + 1]:= symbol; i:= i + 2; s:= s + 1;
            goto string
         end
      end
   end
   else
   begin buffer[i]:= 127; i:= i + 1; goto start1 end
end
else if symbol = 126 then
begin comment _;
   symbol:= undsym;
   if symbol = 70 ∨ symbol = 72 ∨ symbol = 74 ∨ symbol = 76 ∨
   symbol = 90 then
   begin comment =, ≤ , ≥ , ¬ , : ;
      buffer[i]:= 93; buffer[i + 1]:= 126;
      buffer[i + 2]:= symbol; buffer[i + 3]:= 93; i:= i + 4;
      s:= s + 3; goto start
   end
   else
   begin stock[1]:= 126; stock[2]:= symbol; stock[3]:= RESYM;
      for a:= 4 step 2 until 18 do
      begin stock[a]:= undsym; symbol:= stock[a + 1]:= RESYM;
         if symbol ≠ 126 then
         begin k:= a; goto again end
      end;
   again: if (stock[2] = 15 ∧ stock[4] = 10) ∨ (stock[2] = 29
      ∧ stock[4] = 27) then
      begin comment false, true;
         stockbuffer(0, 0, k)
      end
      else if stock[2] = 11 ∧ stock[4] = 14 then
      begin comment begin;
         if i > pos[b] then
         begin for a:= i - 1 step - 1 until i - pos[b] do if
            buffer[a] ≠ 93 then
            begin c:= a;
               if buffer[c] = 90 ∧ c < pos[b] then goto labeg
               else goto pun
            end
         end;
      pun: if i ≠ 1 then punchbuffer;
      labeg: comm:= 1; stockbuffer(0, 1, k); comm:= 0;
         if proc = 0 then
         begin pointer:= pointer + 1;
            proclevel[pointer]:= level; proc:= 1
         end;
         level:= level + 1; b:= if level < n then level else n;
         boolb:= true; tabstop:= tabstop + pos[b];
         w:= pos[b] - 6; if w > 0 then
         begin for a:= 0 step 1 until w - 1 do buffer[i + a]:=
            93; i:= i + w; s:= s + w
         end;
```

```
      goto start1
   end
   else if stock[2] = 13 ∨ (stock[2] = 28 ∧ stock[4] = 29 ∧
   stock[6] = 14) ∨ stock[2] = 32 ∨ stock[2] = 29 ∨
   stock[2] = 30 then
   begin comment do, step, while, then, until;
      stockbuffer(1, 1, k)
   end
   else
e: if stock[2] = 14 ∧ stock[4] = 21 then
   begin comment else;
      if zz = 0 then stockbuffer(1, 1, k) else
      begin zz:= 0; stockbuffer(0, 1, k) end
   end
   else if stock[2] = 14 then
   begin comment end;
      if buffer[1] = 126 ∧ buffer[2] = 11 ∧ buffer[4] = 14
      ∧ s < width - 5 ∧ zz = 0 then zz:= 1 else zz:= 0;
      if zz = 1 then
      begin boolb:= false; goto goon end;
      if i ≠ 1 then punchbuffer;
   goon: comm:= 1;
      if zz = 0 then stockbuffer(0, 0, k) else
      stockbuffer(1, 0, k); comm:= 0;
      tabstop:= tabstop - pos[b]; level:= level - 1;
      if level = 0 then
      begin punchbuffer; sym(119); RUNOUT; nop:= nop + 1;
         if nop = numofpr then goto end else
         begin k:= page + rest - 1;
            for a:= line step 1 until k do sym(119);
            goto repeat
         end
      end
      else b:= if level < n then level else n;
      if symbol = 93 ∨ symbol = 118 ∨ symbol = 119 then
      begin
      reject: symbol:= RESYM;
         if symbol = 93 ∨ symbol = 118 ∨ symbol = 119 then
         goto reject else if symbol = 126 then
         begin stock[1]:= 126;
            for a:= 2 step 2 until 18 do
            begin stock[a]:= undsym;
               symbol:= stock[a + 1]:= RESYM;
               if symbol ≠ 126 then
               begin k:= a; goto nextund end
            end;
         nextund: if stock[2] = 14 then
            begin if i ≠ 1 then punchbuffer; zz:= 1; goto e
            end
            else
            begin if zz = 1 then
               begin breakbuffer; zz:= 0 end;
               buffer[i]:= 93; i:= i + 1; s:= s + 1;
               boolc:= true; goto comm1
            end
```

```
        end
        else if symbol = 91 then semicolon else
        begin if zz = 1 then
           begin breakbuffer; zz:= 0 end;
           buffer[i]:= 93; buffer[i + 1]:= symbol;
           i:= i + 2; s:= s + 2; boolc:= true; goto comm2
        end
     end
     else if symbol = 91 then semicolon else
     begin if zz = 1 then
        begin breakbuffer; zz:= 0 end;
        buffer[i]:= symbol; i:= i + 1; s:= s + 1;
        goto comm2
     end
end
else if stock[2] = 12 then
begin comment comment;
comm1: comm:= 1; stockbuffer(0, 1, k); comm:= 0;
nospace: if symbol = 93 then
   begin symbol:= RESYM; goto nospace end;
   buffer[i]:= symbol; i:= i + 1; s:= s + 1;
comm2: symbol:= RESYM;
   if symbol ≠ 118 ∧ symbol ≠ 119 ∧ symbol ≠ 93 then
   buffer[i]:= symbol else if buffer[i - 1] ≠ 93 then
   buffer[i]:= 93 else goto comm2; if symbol ≠ 91 then
   begin i:= i + 1;
      if symbol ≠ 126 then s:= s + 1 else if boolc then
      begin buffer[i]:= symbol:= undsym; i:= i + 1;
         s:= s + 1; if symbol ≠ 14 then goto comm2 else
         begin stock[1]:= 126; stock[2]:= 14;
            symbol:= stock[3]:= RESYM;
            if symbol = 126 then
            begin for a:= 4 step 2 until 10 do
               begin stock[a]:= undsym;
                  symbol:= stock[a + 1]:= RESYM;
                  if symbol ≠ 126 then
                  begin k:= a; i:= i - 3; s:= s - 2;
                     if stock[4] = 21 ∨ stock[4] = 23
                     then
                     begin i:= i + 1; punchbuffer;
                        boolc:= false;
                        if stock[4] = 21 then zz:= 1;
                        goto e
                     end
                     else
                     begin comm:= 1; stockbuffer(0, 0, k);
                        comm:= 0; goto comm2
                     end
                  end
               end
            end
            else
            begin comm:= 1; stockbuffer(0, 0, 2);
               comm:= 0; buffer[i]:= symbol; goto comm2
            end
```

```
            end
         end;
         if s ≥ width then breakbuffer; goto comm2
      end
      else
      begin i:= i + 1; punchbuffer;
         if level = proclevel[pointer] then
         begin RUNOUT; newline; pointer:= pointer - 1;
            proc:= if pointer = 0 then - 1 else 1
         end;
         boolc:= false; goto start
      end
   end
   else if stock[2] = 27 ∨ (stock[2] = 18 ∧ stock[4] = 23)
   ∨ stock[2] = 11 ∨ stock[2] = 38 then
   begin comment real, integer, boolean, Boolean;
      if stock[k + 1] = 93 then
      begin for a:= k + 2 step 1 until 100 do
         begin symbol:= RESYM; if symbol ≠ 93 then goto dec
         end;
      dec: if symbol = 126 then
         begin stock[k + 2]:= 126;
            for a:= k + 4 step 2 until k + 20 do
            begin stock[a - 1]:= undsym;
               symbol:= stock[a]:= RESYM;
               if symbol ≠ 126 then
               begin h:= a; goto arproc end
            end;
         arproc: if stock[k + 3] = 10 then
            begin comment <type> array;
               decl:= 1; stock[h]:= 93; stockbuffer(0, 0, h)
            end
            else
            begin comment <type> procedure;
               if i ≠ 1 then punchbuffer; if proc ≠ 0 then
               begin RUNOUT; newline; proc:= 0 end;
               decl:= 1; stock[h]:= 93; stockbuffer(0, 0, h)
            end
         end
         else
         begin stock[k + 2]:= symbol;
            symbol:= stock[k + 3]:= RESYM; decl:= 1;
            stockbuffer(0, 0, k + 2)
         end
      end
      else
      begin decl:= 1; stockbuffer(0, 1, k) end
   end
   else if stock[2] = 25 then
   begin comment procedure;
      if proc ≠ 0 then
      begin if i ≠ 1 then punchbuffer; RUNOUT; newline;
         proc:= 0
      end;
      decl:= 1; stockbuffer(0, 1, k)
```

```
                end
                else if stock[2] = 15 then
                begin comment for;
                   if i > pos[b] then
                   begin for a:= i - 1 step - 1 until i - pos[b] do if
                      buffer[a] ≠ 93 then
                      begin c:= a; goto nonl2 end;
                   nonl2: if c > 10 then
                      begin if (buffer[c] = 23 ∧ buffer[c - 1] = 126 ∧
                         buffer[c - 2] = 18) ∨ buffer[c] = 90 then goto
                         lafor
                      end;
                      if buffer[c] = 90 then goto lafor
                   end;
                   if i ≠ 1 then punchbuffer;
                lafor: stockbuffer(0, 1, k)
                end
                else if stock[2] = 24 ∨ (stock[2] = 28 ∧ stock[4] = 29)
                ∨ stock[2] = 10 ∨ stock[2] = 28 ∨ stock[2] = 21 ∨
                stock[2] = 31 then
                begin comment own, string, array, switch, label, value;
                   decl:= 1; stockbuffer(0, 1, k)
                end
                else stockbuffer(0, 1, k)
             end
          end
          else
          begin buffer[i]:= symbol; i:= i + 1; s:= s + 1; goto start end;
    end: STOPCODE
    end
end
```

Appendix I

1. integer procedure RESYM;
   There is no sense in a call of the function designator RESYM if the input-tape is not a 7-holes tape, punched in MC-flexowritercode. RESYM delivers a value which is the internal representation of the first flexowriter-symbol on the tape, different from
   (i) the punchings blank(0), erase(127), stopcode(11) and back-space(42), which are skipped by RESYM,
   (ii) the punchings lower case(122) and upper case(124) with due observance of the last case-definition used by RESYM or READ.
   The internal representation of flexowriter-symbols is given in Appendix II.

2. real procedure READ;
   READ delivers the value of the next number of a data tape.

3. procedure PRSYM(n); value n; integer n;
   PRSYM(n) prints the symbol corresponding to the value of n as given in Appendix II.

4. procedure PUSYM(n); value n; integer n;
   PUSYM is the punching analogon of PRSYM.

5. procedure SPACE(n); value n; integer n;
   SPACE(n) prints n spaces.

6. procedure PUSPACE(n); value n; integer n;
   PUSPACE is the punching analogon of SPACE.

7. procedure PRINTTEXT(s); string s;
   PRINTTEXT(s) prints the symbols of the string s, one after another, not printing the outer stringquotes.

8. procedure NLCR;
   NLCR brings about a line feed and a carriage return.

9. procedure PUNLCR;
   PUNLCR is the punching analogon of NLCR.

10. procedure RUNOUT;
    RUNOUT punches 20 cm. blank tape (= 80 punchings).

11. procedure STOPCODE;
    STOPCODE punches the MC-flexowriter stopcode, followed by RUNOUT.

12. procedure EXIT;
    EXIT finishes the execution of the program.

Appendix II. The internal representation of symbols.

int.repr.	symbol	int.repr.	symbol	int.repr.	symbol
0	0	30	u	61	Y
1	1	31	v	62	Z
2	2	32	w	64	+
3	3	33	x	65	-
4	4	34	y	66	×
5	5	35	z	67	/
6	6	37	A	70	=
7	7	38	B	72	<
8	8	39	C	74	>
9	9	40	D	76	¬
10	a	41	E	79	∨
11	b	42	F	80	∧
12	c	43	G	87	,
13	d	44	H	88	.
14	e	45	I	89	10
15	f	46	J	90	:
16	g	47	K	91	;
17	h	48	L	93	space
18	i	49	M	98	(
19	j	50	N	99	)
20	k	51	O	100	[
21	l	52	P	101	]
22	m	53	Q	118	tabulation ')
23	n	54	R	119	carriage return
24	o	55	S	120	'
25	p	56	T	121	"
26	q	57	U	122	?
27	r	58	V	126	_
28	s	59	W	127	|
29	t	60	X		

') In the MC-ALGOL 60 system a tabulation raises the position on the line by at least 2 and at most 9, in such a way that the position becomes a multiple of eight. In the ALGOL editor, tabulations are used only at the beginning of the line; hence, the result will always be equivalent to 8 spaces.

References:

[1] W.M. McKeeman, Algol 60 Reference language editor, CACM 8(1965), 667-668.

[2] F.E.J. Kruseman Aretz, Het MC-ALGOL 60 systeem voor de X8, MR 81, Mathematisch Centrum, Amsterdam, 1966 (Dutch).