RA

stichting
mathematisch
centrum

MC

REKENAFDELING RA NR 18/71 SEPTEMBER

J.V.M. VAN DER GRINTEN en D. GRUNE
HANDLEIDING BIJ HET PROCEDUREPAKKET
'BASIC SYMBOLS'

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

1. Doel.

Het procedurepakket 'basic symbols' maakt het de ALGOL 60-programmeur mogelijk uit een ALGOL 60-programmatekst symbolen op te bouwen op een manier die identiek is aan die van de ALGOL 60-compiler in het milli-systeem. Vele procedures zijn letterlijke ALGOL 60-vertalingen van overeenkomstige procedures uit de ALGOL 60-compiler. De procedures kunnen symbolen in beide representatiestijlen verwerken. Geen der procedures voert zelf invoeropdrachten uit; sommige procedures gebruiken de regeldrukker.

2. Inleiding.

Men kan zich voorstellen dat de opbouw van basic symbols uit een gegeven ALGOL 60-tekst in drie stadia geschiedt.
Eerst wordt uit de invoer, die bij voorbeeld met resym gelezen wordt, de tekst gedestilleerd zoals deze op de regeldrukker afgedrukt wordt; voor programma's in flexowritercode is hiermee geen enkele verandering gemoeid, uit programma's in ISO-code wordt de CR verwijderd en de LF vervangen door een terug-wagen-nieuwe-regel, terwijl bij programma's op kaarten de laatste acht kolommen overgeslagen worden en het dollarteken verwerkt wordt. De tekst bestaat dan uit 'programmasymbolen'. Verder wordt in dit stadium een regeltelling bijgehouden.

Vervolgens worden in deze tekst waar mogelijk symbolen samengenomen tot ALGOL 60-symbolen, zoals deze vermeld staan in het Revised Report. Zo wordt bij voorbeeld uit de symbolenrij 'times' het ALGOL 60-symbool × opgebouwd, uit de symbolenrij ×× het ALGOL 60-symbool ↑, enz.. Dit proces wordt op elk symbool hoogstens eenmaal toegepast, zodat de symbolenrij 'times'× niet tot het symbool ↑ omgevormd wordt. De tekst bestaat nu uit 'interne symbolen'. Verder wordt in dit stadium een administratie aangaande stringquotes bijgehouden.

Tenslotte wordt uit de zo ontstane ALGOL 60-symbolenstroom alles verwijderd wat niet voor de betekenis van de ALGOL 60-tekst van belang is: spaties, tabs en nlcr's worden verwijderd, evenals commentaar achter end en comment. Parameter-delimiters van de vorm

)<letterstring>:(

(de zogenaamde vette komma's) worden door komma's vervangen, terwijl het symbool e op de daartoe strekkende plaatsen door het symbool ₁₀ vervangen wordt. De tekst bestaat nu uit 'basissymbolen'.

In het bovenstaande wordt de onjuiste indruk gewekt dat deze drie processen achtereenvolgens en afzonderlijk op de gehele tekst toegepast worden. In werkelijkheid worden de processen, door drie integer procedures, op de volgende wijze uitgevoerd: Om een programmasymbool (of eigenlijk diens integer representatie) te verkrijgen kan men de function designator 'nxt prog sbl(resym)' gebruiken. De procedure 'nxt prog sbl' evalueert zijn parameter (d.w.z. resym) zo vaak tot voldoende informatie is verkregen om een programmasymbool af te kunnen leveren. De function designator 'nxt prog sbl(resym)' kan zelf gebruikt worden als parameter van de procedure 'in sbl', welke interne symbolen aflevert: 'in sbl(nxt prog sbl(resym))'. De procedure 'in sbl' evalueert 'nxt prog sbl(resym)' zo vaak als nodig is om een intern symbool af te kunnen leveren. Van eventueel te veel verwerkte symbolen wordt automatisch een administratie bijgehouden (b.v. om het einde van een onderstreept woord te vinden moet een symbool te veel

opgevraagd worden; dit symbool wordt bewaard en bij de volgende aanroep van 'in sbl' als eerste verwerkt).
De function designator 'in sbl(nxt prog sbl(resym))' kan op zijn beurt weer dienen als parameter van de procedure 'nxt bsc sbl', welke basissymbolen aflevert: 'nxt bsc sbl(in sbl(nxt prog sbl(resym)))'. Ook hier wordt de parameter weer zo vaak uitgewerkt (en daarmee 'in sbl' aangeroepen die zijn programmasymbolen verkrijgt door 'nxt prog sbl' aan te roepen, enz.) als nodig is om het volgende basissymbool samen te stellen.

Uiteraard kunnen ook andere constructies als parameter gebruikt worden, mits deze bruikbare symboolwaarden afleveren (zie het voorbeeld in 7.2.).
De programmatekst kan en passant op de regeldrukker afgedrukt worden door de function designator 'nxt prog sbl(resym)' te vervangen door 'print sbl(nxt prog sbl(resym))'. Door op soortgelijke wijze een gebruikersprocedure in de geneste aanroep te betrekken kan de programmeur beschikken over de symbolen terwijl ze van procedure naar procedure doorgegeven worden.

Als tijdens de opbouw van een symbool een ongedefinieerde situatie ontstaat, geeft de betreffende procedure een foutmelding op de regeldrukker, die wat formaat en foutnummer aangaat vrijwel identiek is aan een foutmelding van de ALGOL 60-compiler.

3. De conversieprocedures.

Indien bij een van de onderstaande procedures de parameter de integer representatie van een symbool aflevert dat in dit stadium nog niet of niet meer had mogen voorkomen, wordt dit symbool gewoon doorgegeven (of geskipt na comment of end), tenzij de waarde negatief of ≥ 256 is; in dat geval is het effect ongedefinieerd.

3.1. integer procedure nxt prog sbl(resym); integer resym;

Deze procedure skipt CR-symbolen en vervangt LF-symbolen door nlcr-symbolen; bij kaartinvoer worden de kolommen 73-80 geskipt en het $-symbool verwerkt. De programmatekst heeft nu de vorm die tijdens een compilatie op de regeldrukker verschijnt (zie 4.1.). Verder houdt de procedure een regelnummer bij, dat de programmeur ter beschikking staat als de globale variabele 'line counter'.

3.2. integer procedure in sbl(nxt prog sbl); integer nxt prog sbl;

De procedure bouwt uit opeenvolgende 'nxt prog sbl'-s symbolen in de interne representatie op, zoals vermeld in de tabel in 6.1.1. in LR1.1 (met uitzondering van CR en LF). Hierbij kan een foutmelding met foutnummer 108 optreden (zie 7.2. uit LR1.1).

Voor de eerste stringquote in een string wordt de interne waarde 102 afgeleverd, maar tekst die op deze stringquote volgt wordt alleen dan door 'in sbl' als string geinterpreteerd als het hoofdprogramma de globale variabele 'quote counter' op 1 zet. Het hoofdprogramma moet dus bevestigen dat met deze stringquote inderdaad een string begint. Dat dit noodzakelijk is, blijkt uit situaties als

end ∉ else

en ook uit de gevolgen van een ponsfout als in

if n ∉ 0 then i.p.v. if n ≤ 0 then

waarbij het de voorkeur verdient de rest van het programma niet als string op te vatten.

Heeft 'quote counter' eenmaal een waarde groter dan nul, dan wordt de verdere telling van de quotes door 'in sbl' bijgehouden. Van inner-stringquotes worden de samenstellende delen los afgeleverd, terwijl de laatste unquote als interne waarde 103 verschijnt. Het hoofdprogramma moet ook nu weer het einde van de string bevestigen door de globale variabele 'quote counter' op 0 te zetten.

3.3. integer procedure nxt bsc sbl(in sbl); integer in sbl;

De procedure verwijdert spaties, tabs en nlcr's buiten strings, verwijdert commentaar, zowel na comment als na end, en zet de 'vette komma' om in een gewone komma. Hierbij kunnen foutmeldingen optreden met foutnummers 100 en 101 (zie 7.2. in LR1.1).

Verder wordt binnen getallen het symbool e door het symbool $_{10}$ vervangen.

4. Hulpprocedures.

4.1. integer procedure print sbl(nxt prog sbl);
integer nxt prog sbl;

Deze procedure evalueert de parameter en drukt het overeenkomstige symbool op de regeldrukker af. Is dat symbool een nlcr, dan wordt een regelnummer afgedrukt in een formaat identiek aan dat van de ALGOL 60-compiler; 'print sbl' neemt hierbij aan dat het juiste (nieuwe) regelnummer al in 'line counter' staat en houdt zelf geen regeltelling bij. 'print sbl' levert als waarde de integer representatie van het afgedrukte symbool af.

4.2. procedure print5(n); value n; integer n;

Deze procedure drukt de waarde van n af in 5 cijfers zonder voorafgaande of volgende spatie; non-significante nullen worden vervangen door spaties. De procedure wordt gebruikt voor het afdrukken van regelnummers door 'print sbl'.

4.3. procedure errorm(n); value n; integer n;

De procedure drukt een foutmelding met foutnummer n op de regeldrukker af, in het gebruikelijke formaat. De vermeldingen <laatst gelezen symbool> en <laatst gelezen identifier> ontbreken echter.

4.4. procedure new sbl set;

Deze procedure initialiseert een aantal globale variabelen opnieuw, nadat de conversieprocedures reeds gebruikt zijn. De procedure kan gebruikt worden om in een en hetzelfde programma meerdere programmateksten, eventueel in verschillende stijlen, te verwerken.

5. Globale variabelen.

De volgende globale variabelen staan ter beschikking van de programmeur; ze kunnen alle uitgelezen worden en aan sommige mag geassigneerd worden.

5.1. integer line counter.
bij uitlezen: nummer van de onderhavige regel
bij assignment: het regelnummer krijgt de geassigneerde waarde.

5.2. integer quote counter.
bij uitlezen: quote counter = 0 betekent:
buiten string
quote counter > 0 betekent:
binnen string
bij assignment: quote counter:= 1 betekent:
hier begint een string
quote counter:= 0 betekent:
hier eindigt een string

5.3. integer style.
bij uitlezen: style = 1 betekent:
tekst in onderstreepstijl
style = 0 betekent:
stijl onbekend
style = -1 betekent:
tekst in apostrofstijl
bij assignment: zie boven.

5.4. integer last symbol.
bij uitlezen: laatste door nxt bsc sbl afgeleverde symbool
bij assignment: ongedefinieerd.

5.5. Boolean card format.
Als de invoer in kaartformaat is, en de procedure 'nxt prog sbl' gebruikt wordt, moet voorafgaand aan de eerste aanroep' de statement:

card format:= true

uitgevoerd worden.

6. Gebruik.

Het procedurepakket 'basic symbols' bestaat uit een begin, een aantal declaraties en een statement ter initialisering van een aantal variabelen. Het gebruikersprogramma moet dus met een extra end afgesloten worden.

7. Voorbeelden.

7.1. Het blok

```
begin   integer lst;
        lst:= last symbol;
        n:= nxt bsc sbl(in sbl(print sbl(nxt prog sbl(resym))));
        if n = 102 then
        begin   if lst = 87 ∨ lst = 98
                then quote counter:= 1
                else errorm(200)
        end ≮ else
        if n = 103 then quote counter:= 0;

        comment     value     meaning
                      87          ,
                      98          (
                     102          ≮
                     103          ≯  ;
end
```

levert de waarde van het eerstvolgende betekenisvolle ALGOL 60-symbool in n af, waarbij strings alleen erkend worden als ze voorafgegaan worden door een haakje-open of een komma; stringquotes op andere betekenisvolle plaatsen geven een foutmelding met foutnummer 200.

7.2. Het gebruikersprogramma

```
begin   integer n, save;

        integer procedure alg sym;
        alg sym:= in sbl(nxt prog sbl(resym));

        integer procedure in sbl or save;
        if save > 0 then
        begin in sbl or save:= save; save:= -1 end else
        in sbl or save:= alg sym;

        save:= -1;
next:   n:= nxt bsc sbl(in sbl or save);
        if n = 102 then quote counter:= 1 else
        if n = 103 then quote counter:= 0 else
        if n > 9 ∧ n < 63 ∧ quote counter = 0 then
        begin   comment found identifier;
                nlcr;  print5(line counter);   tab;
        rep:    prsym(n);   n:= alg sym;
                if n < 63 ∨ n = 93 ∨ n = 118 ∨ n = 119
                then goto rep;
                comment 93: space, 118: tab, 119: nlcr;
                save:= n
        end identifier;

        goto next
end
end
```

leest een programmatekst en maakt van elk voorkomen van een identifier melding door deze op een nieuwe regel af te drukken, voorafgegaan door het regelnummer van de regel waarin de identifier voorkomt. De tekst wordt met 'nxt bsc sbl' geskipt tot de eerste letter, die dan het begin van een identifier aangeeft; door nu voor het lezen van de identifier de procedure 'in sbl' te gebruiken, wordt de layout binnen de identifier behouden; het als laatste gelezen symbool, dat niet meer tot de identifier behoort, wordt in save bewaard en als eerstvolgend symbool aan de volgende aanroep van 'nxt bsc sbl' aangeboden. Dit laatste is noodzakelijk, aangezien anders een constructie als:

```
proc(par1)letterstring:(par2)
```

fout geanalyseerd wordt.

8. De ALGOL 60-tekst.

```
begin   comment package 'basic symbols':   version 060971.
    constants are used in the following meanings:
    66 ×    75 ≥     91 semicolon   118 tab
    67 /    76 ¬     92 :=          119 nlcr
    68 :    77 =     93 space       120 '
    69 ↑    78 ¬     98 (           121 "
    70 =    80 ↑     99 )           126
    71 ≠    87 ,    100 [           127 ↓
    72 <    88 .    101 ]          134 CR
    73 ≤    89 10   102 ≮          135 LF
    74 ≥    90 :    103 ≯                    ;

  integer stock, stock1, stock2, line counter, last line number,
    last symbol, style, quote counter, stat, column;
  Boolean card format, line empty;
  integer array word delimiter[0:58];

  procedure print5(n);   value n;   integer n;
  begin   integer sum, count;   real x;
    sum:= 0;   count:= 5;
    x:= n/10↑4 + 5↑-6;
  rep:    n:= entier(x);   sum:= sum + n;   count:= count - 1;
    if count = 0 then prsym(n) else
    begin   prsym(if sum = 0 then 93 else n);   x:= (x - n) × 10;
      goto rep
    end
  end print5;

  procedure errorm(n);   value n;   integer n;
  begin   integer save pos;
    save pos:= print pos;   carriage(2);
    printtext(≮er≯); absfixt(3, 0, n); absfixt(8, 0, line counter);
    carriage(2);   space(save pos)
  end errorm;

  procedure new sbl set;
  begin   if line number ≠ 1 ∨ print pos ≠ 0 then newpage;
    last symbol:= last line number:= stock:= stock1:= stock2:= -1;
    style:= quote counter:= line counter:= stat:= column:= 0;
    card format:= false; line empty:= true
  end new sbl set;

  procedure init bsc sbls;
  begin   integer i,count,symbol,previous symbol,word,value;

    i:= count:= word:= word delimiter[0]:= 0;
    previous symbol:= word delimiter[57]:= word delimiter[58]:=
    67 108 863;

    for symbol:= stringsymbol(i,≮
    ar47co33do22eq6ge11gq11gr10gt10if30le91q91s81t8ne7nq7or15
    and16bgn40div4end41eqv13for18imp14int44not12own42val51arra47
    begi40bool45comm33else32equa6equi13fals53goto17grea10impl14
```

```
    inte44labe50less8note7notg9notl11powe5proc48real43step19
    stri46swit49then31time2true52unti20valu51whil21
    *) while symbol ≠ 255,35 do

    begin   i:= i + 1;
      if symbol < 36 then
      begin   if previous symbol < 10 then
        begin   if symbol < 10 then value:= value × 10 + symbol else
          begin   count:= count + 1;
            word delimiter[count]:= word × 64 + value;
            word:= symbol - 9
          end     change-over
        end else
        if symbol < 10 then value:= symbol else
        word:= word × 32 + symbol - 9;
        previous symbol:= symbol
      end meaningful symbol
    end symbol;

    new sbl set
  end init bsc sbls;

  integer procedure nxt prog sbl(resym);   integer resym;
  begin   comment uses the global names
      card format, column, line counter;
    integer symbol;
  skip0:  symbol:= resym;
    if card format then
    begin   column:= column + 1;
      if column = 81 then column:= 0;
      if column > 72 then goto skip0;
      if symbol = 133 then
      begin   skip1: if resym ≠ 119 then goto skip1;
        column:= 0; goto skip0
      end $
    end card format;
    if symbol = 134 then goto skip0;
    if symbol = 119 ∨ symbol = 135 then
    begin   line counter:= line counter + 1;
      symbol:= 119
    end nlcr symbol;
    nxt prog sbl:= symbol
  end nxt prog sbl;

  integer procedure print sbl(nxt prog sbl);   integer nxt prog sbl;
  begin   comment uses the global names
      line counter, print5, line empty, last line number;
    comment 'line empty' and 'last line number' are used to prevent
      the printing of empty lines as these would result from
      excessive trailing blanks;
    integer symbol;
    symbol:= print sbl:= nxt prog sbl;
    if symbol = 119 then
    begin if line empty then carriage(0) else nlcr;
      print5(line counter); tab; line empty:= false
    end nlcr else
```

```
    begin prsym(symbol);
      line empty:= (line empty ∨ last line number ≠ line number) ∧
                   (symbol = 93 ∨ symbol = 118)
    end;

    last line number:= line number
  end print sbl;

  integer procedure nxt bsc sbl(in sbl); integer in sbl;
  if quote counter > 0 then nxt bsc sbl:= last symbol:= in sbl else
  begin   comment uses the global names
      stock2,last symbol,quote counter,errorm;
    integer symbol;

    integer procedure nxt non layout sbl;
    begin   integer symbol;
    repeat: symbol:= in sbl;
      if symbol =  93 then goto repeat;
      if symbol = 118 then goto repeat;
      if symbol = 119 then goto repeat;
      nxt non layout sbl:= symbol
    end nxt non layout sbl;

    if stock2 ≥ 0 then begin symbol:= stock2;   stock2:= -1   end else
  repeat:
    symbol:= nxt non layout sbl;
    if symbol = 97 then
    begin if if last symbol = 91 then true else last symbol = 104 then
      begin   comment comment after begin or semicolon;
      skip0:  if in sbl ≠ 91 then goto skip0;
        goto repeat
      end
    end symbol = comment else
    if last symbol = 105 then
    begin   comment skip after end;
    skip1:  if symbol = 105 then else
      if symbol =  91 then else
      if symbol =  96 then else
      begin   symbol:= in sbl; goto skip1 end
    end end else
    if symbol = 99 then
    begin   comment );
      if quote counter = 0 then
      begin   stock2:= nxt non layout sbl;
        if stock2 > 9 then
        begin   if stock2 < 64 then
          begin
          skip2:  stock2:= nxt non layout sbl;
            if stock2 < 10 then else
            if stock2 > 63 then else goto skip2;
            if stock2 = 90 then stock2:= nxt non layout sbl
                           else errorm(100);
            if stock2 = 98 then stock2:= -1 else errorm(101);
            symbol:= 87
          end letterstring
        end non-digit string
```

```
    end not inside string
  end symbol = );

  if symbol < 10 then stat:= if stat = 1 then 1 else 2 else
  if symbol > 63 then stat:= 0 else
  if if symbol = 14 then stat = 2 else false then
  begin symbol:= 89; stat:= 0 end e for ₁₀ else
  stat:= 1;

 nxt bsc sbl:= last symbol:= symbol
end nxt bsc sbl;

integer procedure in sbl(nxt sym sbl); integer nxt sym sbl;
begin  comment uses the global names
    stock, stock1,style,errorm,quote counter,stat;
  integer symbol;

  procedure skip bars and uls;
  begin  if stock < 0 then stock:= nxt sym sbl;
  skip:  if symbol= stock then
         begin stock:= nxt sym sbl; goto skip end
  end skip bars and uls;

  Boolean procedure nxt del sbl;
  comment tries to read another symbol that belongs to a delimiter.
    If successful then the character is delivered in symbol,
    else the character is delivered in stock;
  if stock > 0 then nxt del sbl:= false else
  begin
  skip space:
    symbol:= nxt sym sbl;
    if style = 1 then
    begin  if symbol ≠ 126 then
      begin  stock:= symbol; nxt del sbl:= false end else
      begin
      und sbl:
        symbol:= nxt sym sbl;
        if symbol = 126 then goto und sbl;
        if symbol =  93 then goto skip space;
        if symbol >  63 then symbol:= 36;
        nxt del sbl:= true
      end
    end underline style else
    begin
      if if symbol < 10 then false else
      symbol < 64 then nxt del sbl:= true else
      begin  if symbol =  93 then goto skip space;
        if symbol = 120 then stock:= nxt sym sbl else
        begin  stock:= symbol; errorm(108) end;
        nxt del sbl:= false
      end not letter
    end apostrophe style
 end nxt del sbl;

 procedure del sbl;
 begin  comment finds first symbol in 'stock',
```

```
      delivers value of delimiter in 'symbol',
      delivers next character in 'stock';
    integer word,i,delta i,test;

    word:= 0;   symbol:= stock;   stock:= -1;
  next1:
    word:= word × 32 +
          (if symbol < 10 then 27 else
      if symbol < 36 then symbol -9 else
      symbol - 36);
    if word < 32768 then
    begin   if nxt del sbl then goto next1 end;

    word:= word × 64; i:= 28; test:= word - word delimiter[28];
    for delta i:= 16,8,4,2,1,0 do
    begin   if test < 0 then
      begin   if test > -64 then goto found;
        i:= i - delta i
      end else
      i:= i + delta i;
      test:= word - word delimiter[i]
    end delta i;

    test:= -98;
  found:  if nxt del sbl then goto found;
    symbol:= 64 - test
  end del sbl;

  if stock > 0 then
  begin   symbol:= stock;
    if stock 1 > 0 then
    begin stock:= stock 1; stock 1:= -1 end
    else stock:= -1
  end stock > 0 else
  symbol:= nxt sym sbl;

  if symbol < 66 then else
  if symbol > 255 then symbol:= symbol - 256 else
  begin   if style= 0 then
    begin   if symbol= 126 then style:= 1 else
      if symbol= 120 then style:= -1
    end set representation;

    if quote counter > 0 then
    begin   if symbol= 127 then
      begin   skip bars and uls;
        if style= 1 then
        begin   if stock= 74 then
          begin   if quote counter= 1 then
            begin symbol:= 103; stock:= - symbol end else
            begin quote counter:= quote counter - 1;
              stock:= stock + 256
            end
          end stock = > else
          if stock= 72 then
          begin   quote counter:= quote counter + 1;
```

```
          stock:= stock + 256
        end stock = <
      end style = underlined
    end symbol= bar else

    if symbol= 120 then
    begin   if style= -1 then
      begin   if stock < 0 then stock:= nxt sym sbl;
        if stock= 99 then
        begin   stock 1:= nxt sym sbl;
          if stock 1= 120 then
          begin   if quote counter= 1 then
            begin   symbol:= 103;
              stock:= stock1:= - symbol
            end
            else
            begin   stock:= stock + 256; stock1:= stock 1 + 256;
              quote counter:= quote counter -1
            end
          end
        end stock= ) else

        if stock= 120 then
        begin   if quote counter= 1 then
          begin symbol:= 103; stock:= - symbol end else
          begin stock:= stock + 256;
            quote counter:= quote counter - 1
          end
        end else
        if stock= 98 then
        begin   stock 1:= nxt sym sbl;
          if stock 1= 120 then
          begin   quote counter:= quote counter + 1;
            stock:= stock + 256;
            stock 1:= stock1 + 256
          end
        end stock= (
      end
    end symbol= accent else

    if symbol= 121 then
    begin   if style= -1 then
      begin   if quote counter= 1 then symbol:= 103
        else quote counter:= quote counter -1
      end
    end symbol = double accent else

    if symbol= 126 then skip bars and uls
  end quote counter > 0 else

  if symbol= 126 then
  begin   skip bars and uls;
    if stock > 63 then
    begin   symbol:= if stock= 90 then 68
          else if stock= 72 then 73
          else if stock= 74 then 75
```

```
          else if stock= 70 then 77
          else if stock= 76 then 78
          else 161;
        stock:= - symbol
      end stock > 63 else

      if style= 1 then del sbl else
      begin symbol:= 161; stock:= - symbol end
    end symbol= underline else

    if symbol= 120 then
    begin   if style= -1 then
      begin   if stock < 0 then stock:= nxt sym sbl;
        if if stock < 64 then stock > 9 else false then del sbl else

        if stock = 98 then
        begin   stock1:= nxt sym sbl;
          if stock1= 120 then
          begin   symbol:= 102; stock:= stock1:= -symbol end
        end stock = ( else
        if stock = 120 then
        begin symbol:= 102; stock:= -symbol end stock = accent else
        if stock = 67 then
        begin   stock1:= nxt sym sbl;
          if stock1 = 120 then
          begin   symbol:= 68; stock:= stock1:= -symbol end
        end stock = /
      end style = apostrophed
    end symbol = accent else

    if symbol = 90 then
    begin   if stock < 0 then stock:= nxt sym sbl;
      if stock = 70 then
      begin   symbol:= 92; stock:= -symbol end
    end symbol = colon else


    if symbol = 127 then
    begin   skip bars and uls;
      symbol:=
        if stock = 80 then 69
        else if stock = 70 then 71
        else if if stock = 72 then style = 1 else false then 102
        else if if stock = 74 then style = 1 else false then 103
        else 160;
      stock:= -symbol
    end symbol = bar else

    if if symbol = 121 then style = -1 else false then
    symbol:= 102 else
    if symbol = 88 then
    begin   if stock < 0 then stock:= nxt sym sbl;
      if stock = 87 then begin symbol:= 91; stock:= -symbol end
      else
      begin   if stock = 88 then
        begin symbol:= 90; stock:= nxt sym sbl end;
```

```
        if stock = 70 then begin symbol:= 92; stock:= -symbol end
      end
    end symbol = period else

    if symbol = 66 then
    begin   if stock < 0 then stock:= nxt sym sbl;
      if stock = 66 then begin symbol:= 69; stock:= -symbol end
    end xx else
    if symbol = 98 then
    begin   if stock < 0 then stock:= nxt sym sbl;
      if stock = 67 then begin symbol:= 100; stock:= -symbol end
    end (/ else
    if symbol = 67 then
    begin   if stock < 0 then stock:= nxt sym sbl;
      if stock = 99 then begin symbol:= 101; stock:= -symbol end
    end /)
  end;

  in sbl:= symbol
end in sbl;

init bsc sbls;
```