On generating documented source code by blocks in TEX

Jan Šustek

Outline

Motivation

Presents

A present will be given to a person who

- finds a preview from word,
- tells the name of a file,
- explains why a colon is used.

Motivation
Wet and dry approach

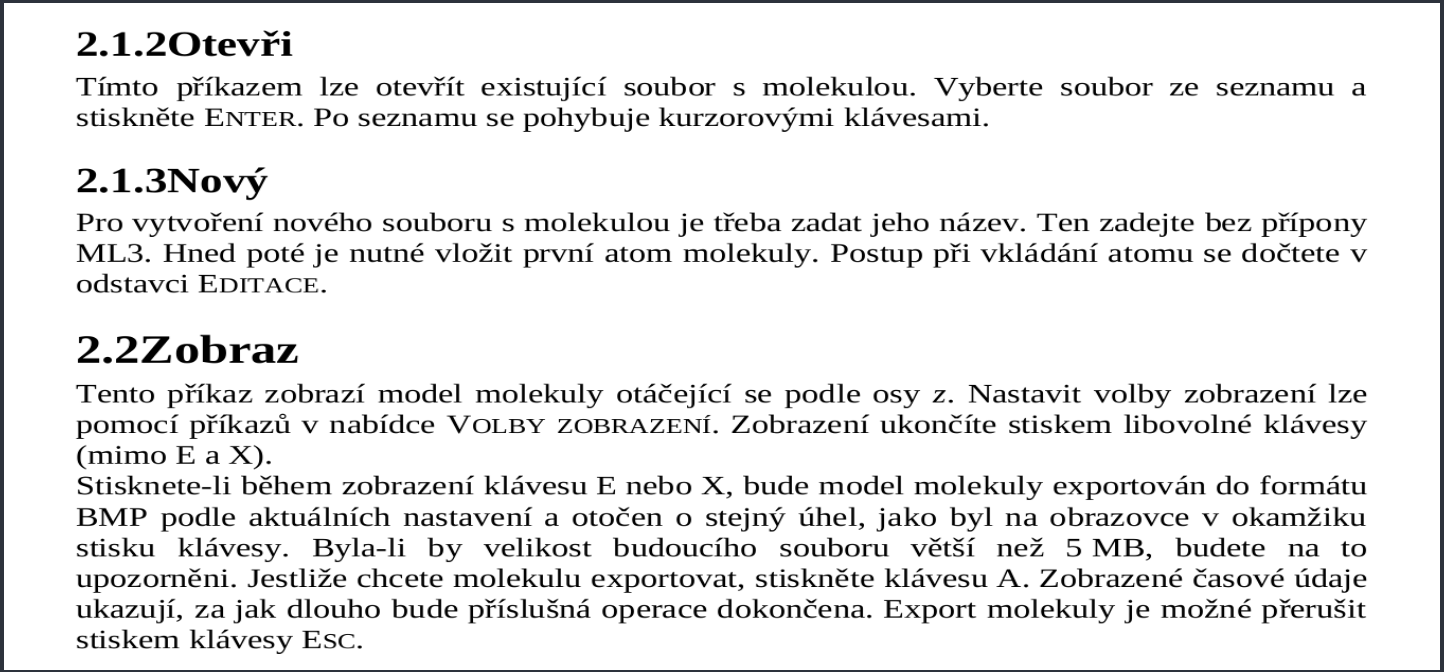
WET = Write Everything Twice

DRY = Don’t Repeat Yourself

Motivation
Program

```
procedure zobraz_;
  var i:word; k:char;
  begin
    new(b);
    init; i:=0;
    repeat
      i:=i+rotace;
      if i>360 then i:=i-360;
      if i<0 then i:=i+360;
      zobraz(i); delay(zdrz);
    until keypressed; k:=readkey;
    if k in ['E','X','e','x'] then
      begin exportuj('molekula'); exit; end;
    closegraph;
    dispose(b);
  end;
```

Motivation
Documentation



2.1.2 Otevři

Tímto příkazem lze otevřít existující soubor s molekulou. Vyberte soubor ze seznamu a stiskněte ENTER. Po seznamu se pohybuje kurzorovými klávesami.

2.1.3 Nový

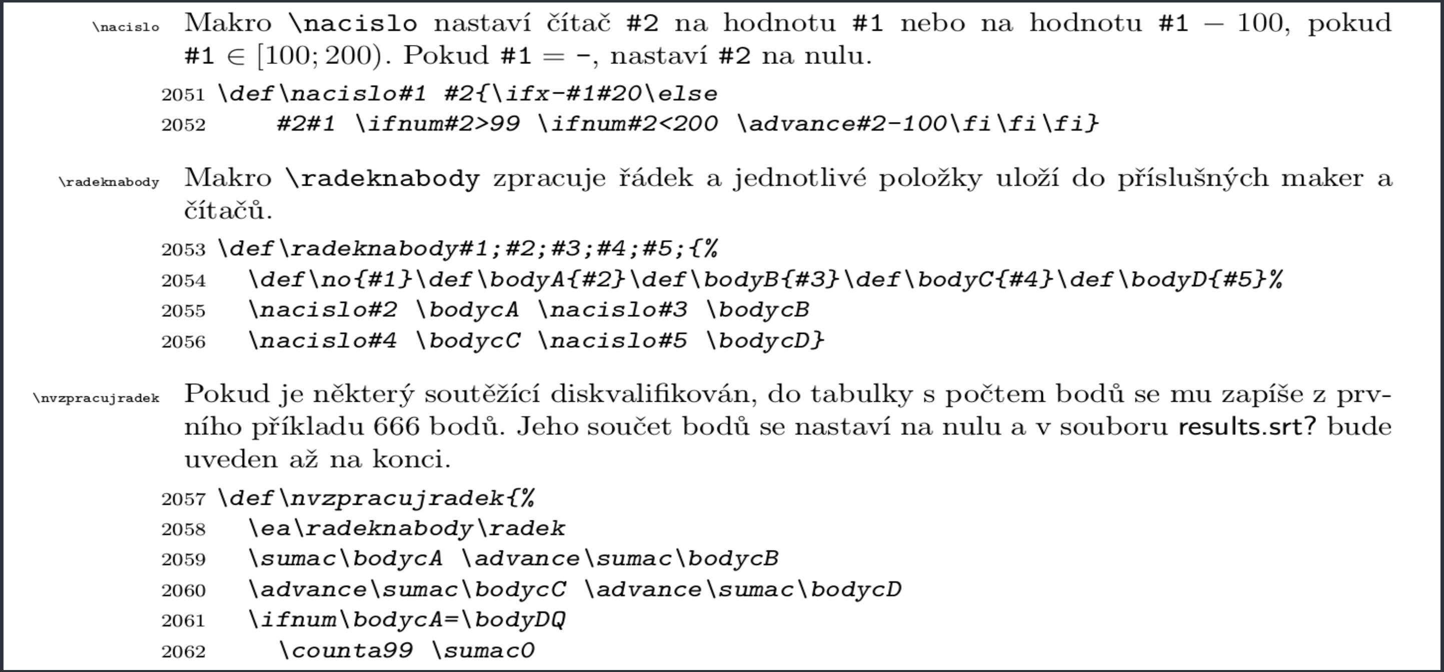
Pro vytvoření nového souboru s molekulou je třeba zadat jeho název. Ten zadejte bez přípony ML3. Hned poté je nutné vložit první atom molekuly. Postup při vkládání atomu se dočtete v odstavci EDITACE.

2.2 Zobraz

Tento příkaz zobrazí model molekuly otáčející se podle osy z. Nastavit volby zobrazení lze pomocí příkazů v nabídce VOLBY ZOBRAZENÍ. Zobrazení ukončíte stiskem libovolné klávesy (mimo E a X).
Stisknete-li během zobrazení klávesu E nebo X, bude model molekuly exportován do formátu BMP podle aktuálních nastavení a otočen o stejný úhel, jako byl na obrazovce v okamžiku stisku klávesy. Byla-li by velikost budoucího souboru větší než 5 MB, budete na to upozorněni. Jestliže chcete molekulu exportovat, stiskněte klávesu A. Zobrazené časové údaje ukazují, za jak dlouho bude příslušná operace dokončena. Export molekuly je možné přerušit stiskem klávesy ESC.

docstrip

Generated documentation



\nacislo

Makro \nacislo nastaví čítač #2 na hodnotu #1 nebo na hodnotu #1 − 100, pokud #1 ∈ [100; 200). Pokud #1 = -, nastaví #2 na nulu.

```
\def\nacislo#1 #2{\ifx-#1#20\else
      #2#1 \ifnum#2>99 \ifnum#2<200 \advance#2-100\fi\fi\fi}
```

\radeknabody

Makro \radeknabody zpracuje řádek a jednotlivé položky uloží do příslušných maker a čítačů.

```
\def\radeknabody#1;#2;#3;#4;#5;{%
  \def\no{#1}\def\bodyA{#2}\def\bodyB{#3}\def\bodyC{#4}\def\bodyD{#5}%
  \nacislo#2 \bodycA \nacislo#3 \bodycB
  \nacislo#4 \bodycC \nacislo#5 \bodycD}
```

\nvzpracujradek

Pokud je některý soutěžící diskvalifikován, do tabulky s počtem bodů se mu zapíše z prvního příkladu 666 bodů. Jeho součet bodů se nastaví na nulu a v souboru results.srt? bude uveden až na konci.

```
\def\nvzpracujradek{%
  \ea\radeknabody\radek
  \sumac\bodycA \advance\sumac\bodycB
  \advance\sumac\bodycC \advance\sumac\bodycD
  \ifnum\bodycA=\bodyDQ
    \counta99 \sumac0
```

docstrip

Generated program

```
\def\nacislo#1 #2{\ifx-#1#20\else
    #2#1 \ifnum#2>99 \ifnum#2<200 \advance#2-100\fi\fi\fi}

\def\radeknabody#1;#2;#3;#4;#5;{%
  \def\no{#1}\def\bodyA{#2}\def\bodyB{#3}\def\bodyC{#4}\def\bodyD{#5}%
  \nacislo#2 \bodycA \nacislo#3 \bodycB
  \nacislo#4 \bodycC \nacislo#5 \bodycD}

\def\nvzpracujradek{%
  \ea\radeknabody\radek
  \sumac\bodycA \advance\sumac\bodycB
  \advance\sumac\bodycC \advance\sumac\bodycD
  \ifnum\bodycA=\bodyDQ
    \counta99 \sumac0
```

docstrip
Source file

```
% Makro |\radeknabody| zpracuje řádek a jednotlivé položky uloží
% do příslušných maker a čítačů.
%    \begin{macrocode}
\def\radeknabody#1;#2;#3;#4;#5;{%
  \def\no{#1}\def\bodyA{#2}\def\bodyB{#3}\def\bodyC{#4}\def\bodyD{#5}%
  \nacislo#2 \bodycA \nacislo#3 \bodycB
  \nacislo#4 \bodycC \nacislo#5 \bodycD}
%    \end{macrocode}
% Pokud je některý soutěžící diskvalifikován, do tabulky s~počtem bodů
% se mu zapíše z~prvního příkladu 666~bodů. Jeho součet bodů se nastaví
% na nulu a v~souboru \soub{results.srt?} bude uveden až na konci.
%    \begin{macrocode}
\def\nvzpracujradek{%
  \ea\radeknabody\radek
...
%    \end{macrocode}
```

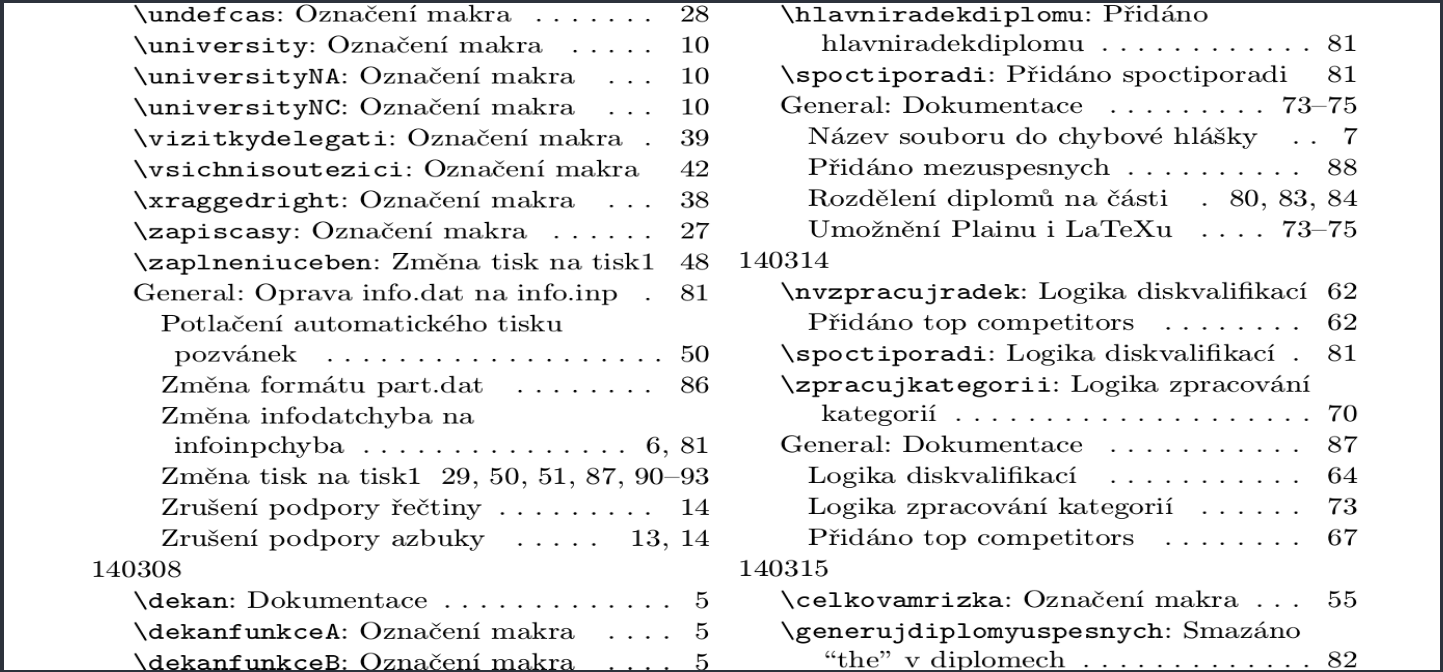
docstrip

Added value

```
% \begin{macro}{\nvzpracujradek}
% Pokud je některý soutěžící diskvalifikován, do tabulky s~počtem bodů
% se mu zapíše z~prvního příkladu 666~bodů. Jeho součet bodů se nastaví
% na nulu a v~souboru \soub{results.srt?} bude uveden až na konci.
% \changes{100908}{100908}{Pročištění}
% \changes{140314}{140314}{Logika diskvalifikací}
% \changes{140314}{140314}{Přidáno top competitors}
%    \begin{macrocode}
\def\nvzpracujradek{%
  \ea\radeknabody\radek
  \sumac\bodycA \advance\sumac\bodycB
  \advance\sumac\bodycC \advance\sumac\bodycD
  \ifnum\bodycA=\bodyDQ
...
%    \end{macrocode}
% \end{macro}
```

docstrip

List of changes



\undefcas: Označení makra 28
\university: Označení makra 10
\universityNA: Označení makra . . . 10
\universityNC: Označení makra . . . 10
\vizitkydelegati: Označení makra . 39
\vsichnisoutezici: Označení makra 42
\xraggedright: Označení makra . . . 38
\zapiscasy: Označení makra 27
\zaplneniuceben: Změna tisk na tisk1 48
General: Oprava info.dat na info.inp . 81
  Potlačení automatického tisku
    pozvánek 50
  Změna formátu part.dat 86
  Změna infodatchyba na
    infoinpchyba 6, 81
  Změna tisk na tisk1 29, 50, 51, 87, 90–93
  Zrušení podpory řečtiny 14
  Zrušení podpory azbuky 13, 14

140308

\dekan: Dokumentace 5
\dekanfunkceA: Označení makra 5
\dekanfunkceB: Označení makra 5

\hlavniradekdiplomu: Přidáno
    hlavniradekdiplomu 81
\spoctiporadi: Přidáno spoctiporadi 81
General: Dokumentace 73–75
  Název souboru do chybové hlášky . . 7
  Přidáno mezuspesnych 88
  Rozdělení diplomů na části . 80, 83, 84
  Umožnění Plainu i LaTeXu 73–75

140314

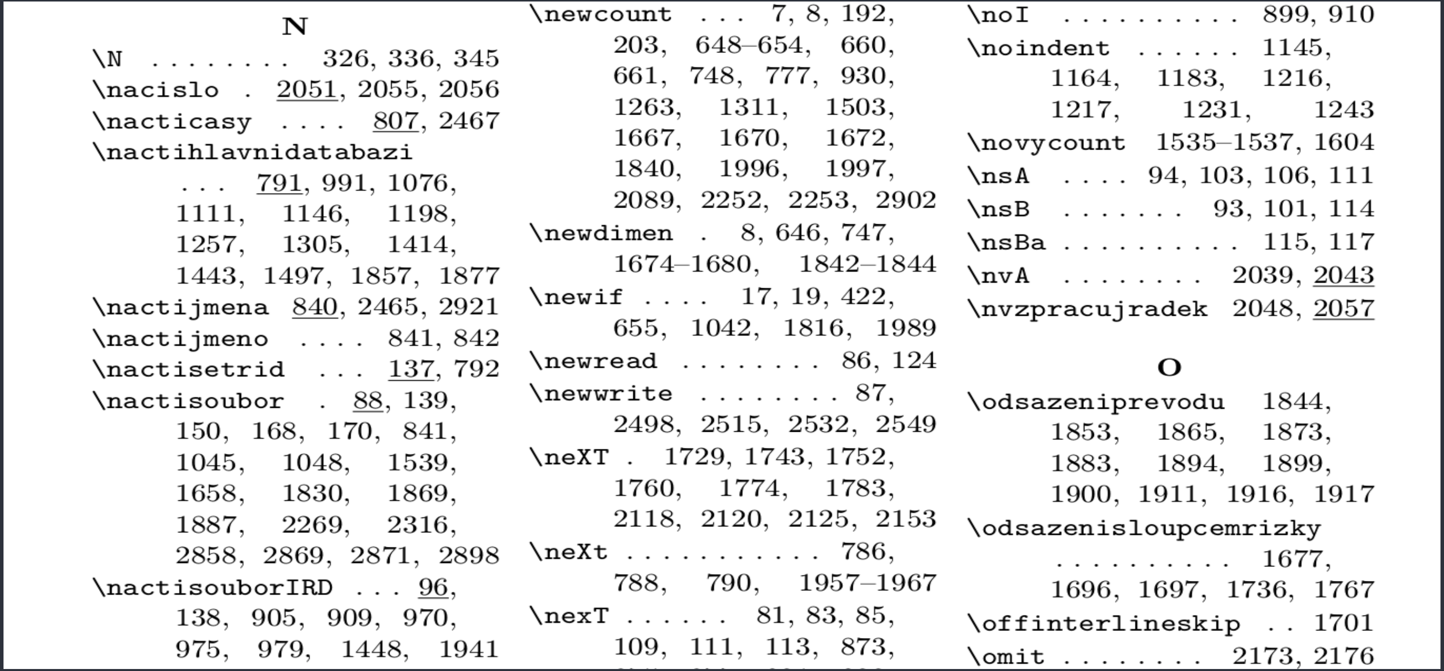
\nvzpracujradek: Logika diskvalifikací 62
  Přidáno top competitors 62
\spoctiporadi: Logika diskvalifikací . 81
\zpracujkategorii: Logika zpracování
    kategorií 70
General: Dokumentace 87
  Logika diskvalifikací 64
  Logika zpracování kategorií 73
  Přidáno top competitors 67

140315

\celkovamrizka: Označení makra . . . 55
\generujdiplomyuspesnych: Smazáno
    “the” v diplomech 82

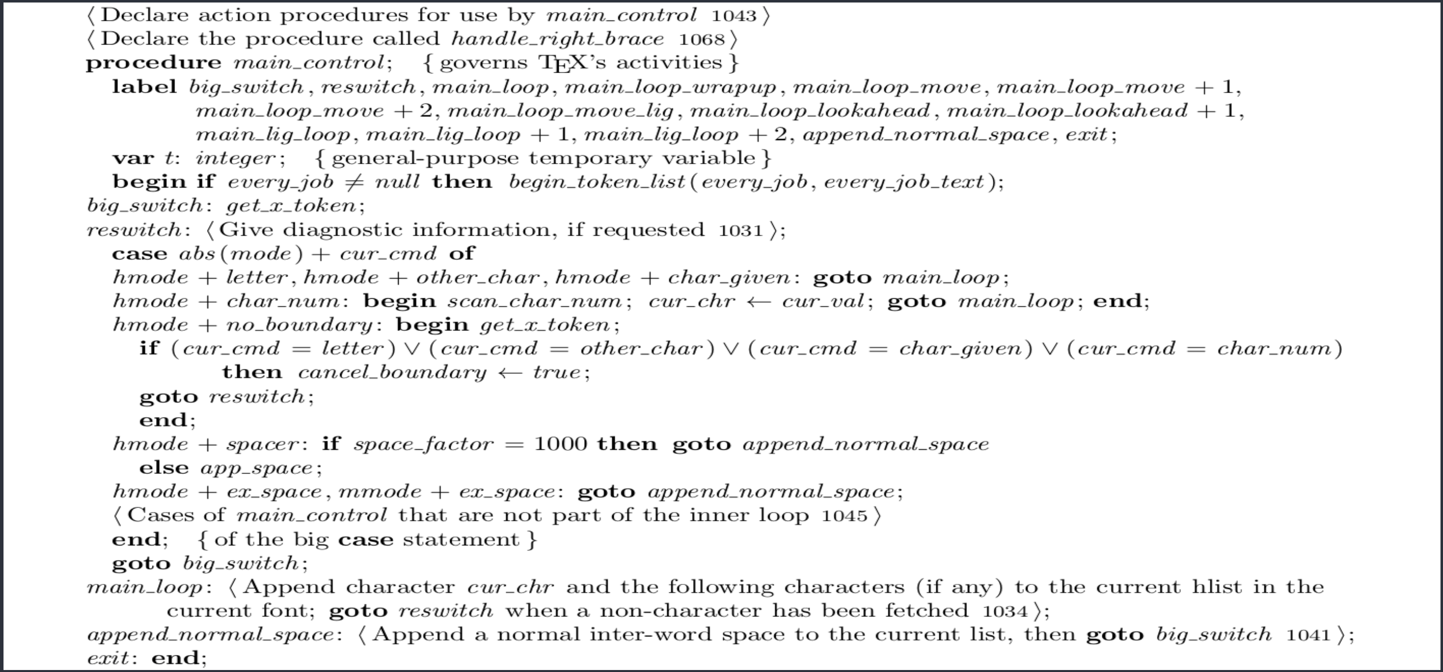
docstrip

Index



DEK and web

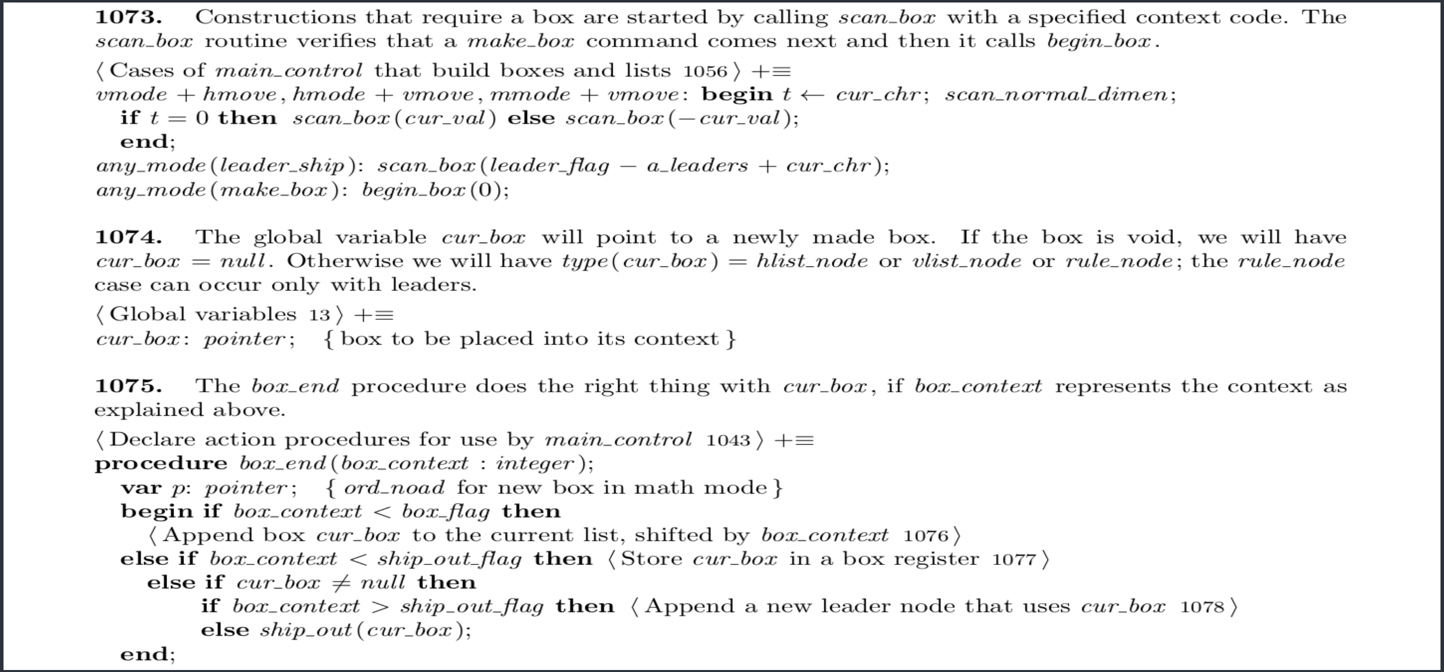
Generated documentation



```
⟨ Declare action procedures for use by main_control 1043 ⟩
⟨ Declare the procedure called handle_right_brace 1068 ⟩
procedure main_control;  { governs TEX's activities }
  label big_switch, reswitch, main_loop, main_loop_wrapup, main_loop_move, main_loop_move + 1,
        main_loop_move + 2, main_loop_move_lig, main_loop_lookahead, main_loop_lookahead + 1,
        main_lig_loop, main_lig_loop + 1, main_lig_loop + 2, append_normal_space, exit;
  var t: integer;  { general-purpose temporary variable }
  begin if every_job ≠ null then begin_token_list(every_job, every_job_text);
big_switch: get_x_token;
reswitch: ⟨ Give diagnostic information, if requested 1031 ⟩;
  case abs(mode) + cur_cmd of
  hmode + letter, hmode + other_char, hmode + char_given: goto main_loop;
  hmode + char_num: begin scan_char_num; cur_chr ← cur_val; goto main_loop; end;
  hmode + no_boundary: begin get_x_token;
    if (cur_cmd = letter) ∨ (cur_cmd = other_char) ∨ (cur_cmd = char_given) ∨ (cur_cmd = char_num)
          then cancel_boundary ← true;
    goto reswitch;
    end;
  hmode + spacer: if space_factor = 1000 then goto append_normal_space
    else app_space;
  hmode + ex_space, mmode + ex_space: goto append_normal_space;
  ⟨ Cases of main_control that are not part of the inner loop 1045 ⟩
  end;  { of the big case statement }
  goto big_switch;
main_loop: ⟨ Append character cur_chr and the following characters (if any) to the current hlist in the
      current font; goto reswitch when a non-character has been fetched 1034 ⟩;
append_normal_space: ⟨ Append a normal inter-word space to the current list, then goto big_switch 1041 ⟩;
exit: end;
```

DEK and web

Generated documentation



1073. Constructions that require a box are started by calling *scan_box* with a specified context code. The *scan_box* routine verifies that a *make_box* command comes next and then it calls *begin_box*.

```
⟨ Cases of main_control that build boxes and lists 1056 ⟩ +≡
vmode + hmove, hmode + vmove, mmode + vmove: begin t ← cur_chr; scan_normal_dimen;
  if t = 0 then scan_box(cur_val) else scan_box(−cur_val);
  end;
any_mode(leader_ship): scan_box(leader_flag − a_leaders + cur_chr);
any_mode(make_box): begin_box(0);
```

1074. The global variable *cur_box* will point to a newly made box. If the box is void, we will have *cur_box* = *null*. Otherwise we will have *type*(*cur_box*) = *hlist_node* or *vlist_node* or *rule_node*; the *rule_node* case can occur only with leaders.

```
⟨ Global variables 13 ⟩ +≡
cur_box: pointer;  { box to be placed into its context }
```

1075. The *box_end* procedure does the right thing with *cur_box*, if *box_context* represents the context as explained above.

```
⟨ Declare action procedures for use by main_control 1043 ⟩ +≡
procedure box_end(box_context : integer);
  var p: pointer;  { ord_noad for new box in math mode }
  begin if box_context < box_flag then
    ⟨ Append box cur_box to the current list, shifted by box_context 1076 ⟩
  else if box_context < ship_out_flag then ⟨ Store cur_box in a box register 1077 ⟩
    else if cur_box ≠ null then
        if box_context > ship_out_flag then ⟨ Append a new leader node that uses cur_box 1078 ⟩
        else ship_out(cur_box);
  end;
```

DEK and web

Generated program

```
{:1070}{1075:}procedure boxend(boxcontext:integer);var p:halfword;
begin if boxcontext<1073741824 then{1076:}begin if curbox<>0 then begin
mem[curbox+4].int:=boxcontext;
if abs(curlist.modefield)=1 then begin appendtovlist(curbox);
if adjusttail<>0 then begin if 29995<>adjusttail then begin mem[curlist.
tailfield].hh.rh:=mem[29995].hh.rh;curlist.tailfield:=adjusttail;end;
adjusttail:=0;end;if curlist.modefield>0 then buildpage;
end else begin if abs(curlist.modefield)=102 then curlist.auxfield.hh.lh
:=1000 else begin p:=newnoad;mem[p+1].hh.rh:=2;mem[p+1].hh.lh:=curbox;
curbox:=p;end;mem[curlist.tailfield].hh.rh:=curbox;
curlist.tailfield:=curbox;end;end;
end{:1076}else if boxcontext<1073742336 then{1077:}if boxcontext<
1073742080 then eqdefine(-1073738146+boxcontext,119,curbox)else
geqdefine(-1073738402+boxcontext,119,curbox){:1077}else if curbox<>0
then if boxcontext>1073742336 then{1078:}begin{404:}repeat getxtoken;
until(curcmd<>10)and(curcmd<>0){:404};
if((curcmd=26)and(abs(curlist.modefield)<>1))or((curcmd=27)and(abs(
curlist.modefield)=1))then begin appendglue;
mem[curlist.tailfield].hh.b1:=boxcontext-(1073742237);
mem[curlist.tailfield+1].hh.rh:=curbox;
end else begin begin if interaction=3 then;printnl(262);print(1065);end;
begin helpptr:=3;helpline[2]:=1066;helpline[1]:=1067;helpline[0]:=1068;
end;backerror;flushnodelist(curbox);end;end{:1078}else shipout(curbox);
end;{:1075}{1079:}procedure beginbox(boxcontext:integer);label 10,30;
```

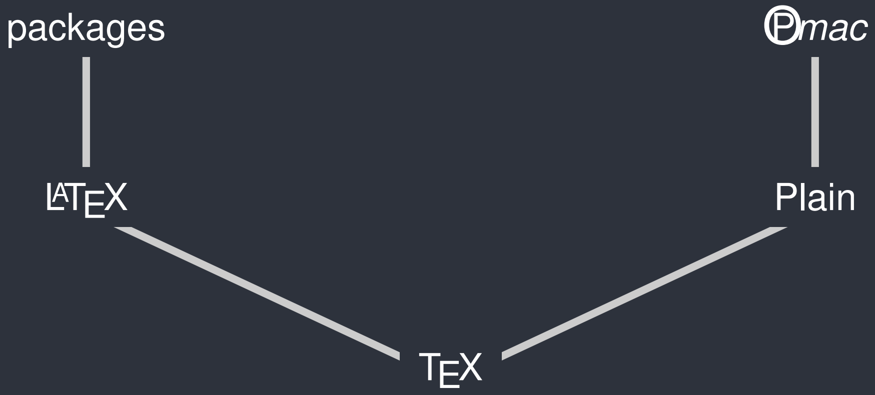
DEK and web
Source file

GIFT

```
@ The |box_end| procedure does the right thing with |cur_box|, if
|box_context| represents the context as explained above.

@<Declare act...@>=
procedure box_end(@!box_context:integer);
var p:pointer; {|ord_noad| for new box in math mode}
begin if box_context<box_flag then @<Append box |cur_box| to the current list,
    shifted by |box_context|@>
else if box_context<ship_out_flag then @<Store \(c)|cur_box| in a box register@>
else if cur_box<>null then
  if box_context>ship_out_flag then @<Append a new leader node that
      uses |cur_box|@>
  else ship_out(cur_box);
end;
```

OpTmac

Map of TeX extensions



OPmac

Source file

```
\input gensrc
\SRCFILENAME style.tex
\activettchar"
Format of the page is~A4 with~2cm~margins.
The basic font size is set to 12\,pt.
\BEGSRC
\margins/1 a4 (2,2,2,2)cm
\typosize[12/14]
\ENDSRC
Macro "\safedef" defines a control sequence which is not yet defined.
If it is already defined then its new definition is ignored.
\BEGSRC
\def\safedef#1{\ifx#1\undefined\else\begingroup\afterassignment\endgroup\fi\def#1}
\ENDSRC
Sections are defined in the same way as in~\OPmac.
\BEGSRC
\safedef\section{\sec}
\ENDSRC
\bye
```

OPmac

Macro \begtt

```
\def\begtt {\par\ttskip\bgroup \wipeepar
  \setverb \adef{ }{ }
  \ifx\savedttchar\undefined \else \catcode\savedttchar=12 \fi
  \parindent=\ttindent \vskip\parskip \parskip=0pt
          \tthook          \relax
  \ifnum\ttline<0 \else
    \tenrm \thefontscale[700]\let\sevenrm=\thefont
    \everypar={\global\advance \ttline by1
      \llap{\sevenrm\the \ttline\kern.9em}}\fi
  \def\par##1{\endgraf\ifx##1\egroup\else
    \penalty\ttpenalty           \leavevmode\fi ##1}
  \obeylines \startverb}
\def\setverb{\frenchspacing\def\do##1{\catcode`##1=12}\dospecials
  \catcode`\*=12 }
{\catcode`\|=0 \catcode`\\=12
|gdef|startverb#1\endtt{|tt#1|egroup|par|ttskip|testparA}}
```

Macro \BEGSRC

```
\def\BEGSRC{\par\ttskip\bgroup \wipeepar
  \setSRC  \adef{ }{ }
  \ifx\savedttchar\undefined \else \catcode\savedttchar=12 \fi
  \parindent=\ttindent \vskip\parskip \parskip=0pt
  \csname SRChook\endcsname\relax
  \ifnum\ttline<0 \else
    \tenrm \thefontscale[700]\let\sevenrm=\thefont
    \everypar={\global\advance\SRCline by1
      \llap{\sevenrm\the\SRCline\kern.9em}\SRCgetline}\fi
  \def\par##1{\endgraf\ifx##1\egroup\else
    \penalty\ttpenalty\expandafter\quitvmode\fi ##1}
  \obeylines \startSRC }
\def\setSRC {\frenchspacing\def\do##1{\catcode'##1=12}\dospecials
  \catcode'\*=12 }
{\catcode'\|=0 \catcode'\\=12
|gdef|startSRC#1\ENDSRC{|tt#1|egroup|par|ttskip|testparA}}
\begingroup\lccode'\~13
  \lowercase{\endgroup\def\SRCgetline#1~}{#1\immediate\write\SRCfile{#1}\par}
\def\SRCFILENAME{\immediate\closeout\SRCfile\immediate\openout\SRCfile=}
\newcount\SRCline \newwrite\SRCfile
```

Simple improvements

Possibility to use `\endSRChook`

```
{\catcode'\|=0 \catcode'\\=12
|gdef|startSRC#1\ENDSRC{|tt#1|egroup|par|ttskip
  |csname endSRChook|endcsname|testparA}}
```

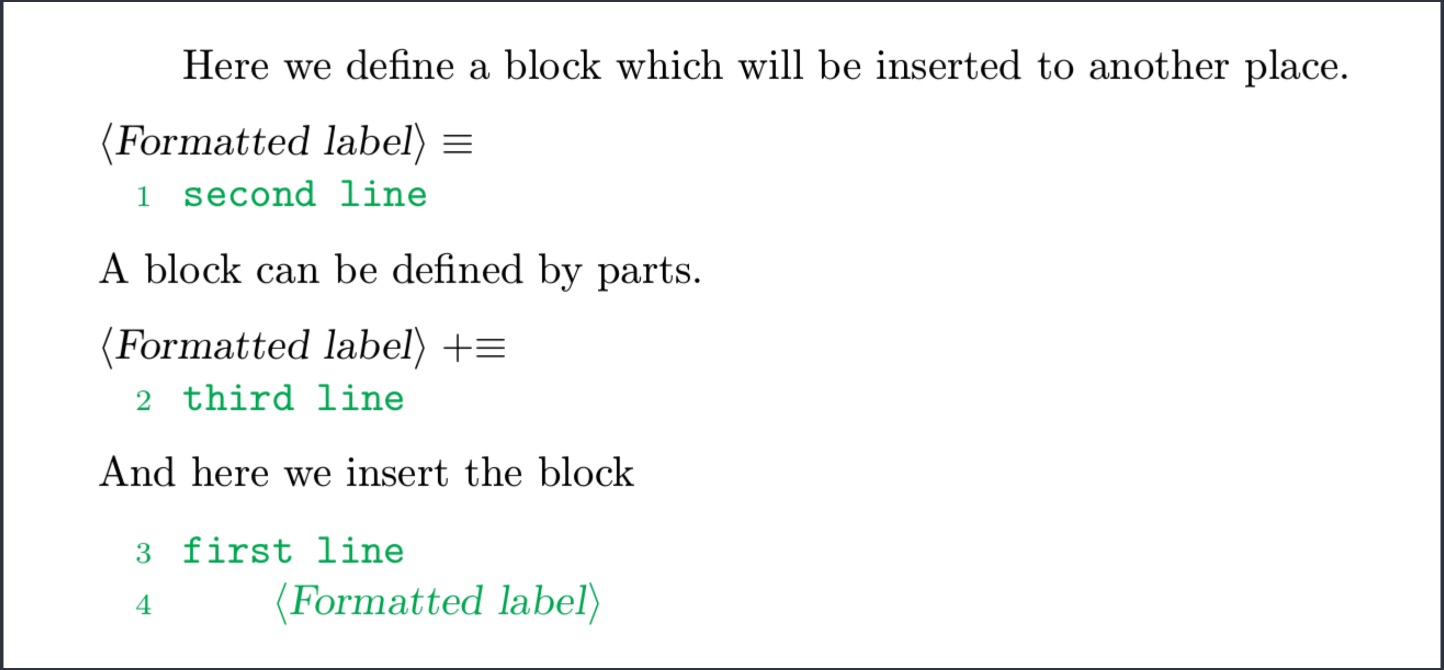
Tabulator expanding to a sequence of spaces

```
{\catcode9=12
  \gdef\setSRC{\def\do##1{\catcode'##1=12}\dospecials
    \catcode'\*=12 \adef{^^I}{\SRCtab}}}
{\catcode32=13 \gdef\SRCtab{    }}
```

Blocks
Source file

```
\input gensrc
\def\SRChook{\longlocalcolor\Green}
\SRCFILENAME program.txt
Here we define a block which will be inserted to another place.
\BEGSRC<InternalLabel>{Formatted label}
second line
\ENDSRC
A block can be defined by parts.
\BEGSRC<InternalLabel>
third line
\ENDSRC
And here we insert the block
\BEGSRC
first line
    |<InternalLabel>
\ENDSRC
\bye
```

Blocks
Generated documentation



Here we define a block which will be inserted to another place.

⟨*Formatted label*⟩ ≡

```
second line
```

A block can be defined by parts.

⟨*Formatted label*⟩ +≡

```
third line
```

And here we insert the block

```
first line
        ⟨Formatted label⟩
```

Blocks

Generated program

```
first line
    second line
    third line
```

Implementation

Macro \SRCgetline

```
\begingroup\lccode`\~13 \lowercase{\endgroup\def\SRCgetline#1~}{%
  \SRCscan#1\<>\SRCend
  \ifx\SRCnaz\relax
    \addto\SRCcontent{\SRConeline{#1}}%
    \SRCprintline{#1}%
  \else
    \expandafter\expandafter\expandafter\addto
      \expandafter\expandafter\expandafter\SRCcontent
      \expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter\SRCblock
        \expandafter\expandafter\expandafter{%
        \expandafter\SRCods\expandafter}\expandafter{\SRCnaz}}%
    \SRCprintblock{\SRCods}{\SRCnaz}%
  \fi\par}
\def\SRCprintline#1{\SRCprintlinenum#1\par}
\def\SRCprintblock#1#2{\SRCprintlinenum
  #1\SRCangle{\csname SRCtit:#2\endcsname}\par}
```

Implementation

Macro \SRCscan

```
\def\SRCnoex{\noexpand}
\def\SRCscan#1\<#2>#3\SRCend{%
  \ifx~#3~%
    \let\SRCnaz\relax
  \else
    \def\SRCods{#1}%
    \ifx\SRCods\SRCnoex
      \def\SRCods{}%
    \fi
    \def\SRCnaz{#2}%
  \fi}
```

Implementation

Tokens \ENDSRC

GIFT

After expansion of

```
\BEGSRC<InternalLabel>{Formatted label}
second line
  |<anotherblock>
\ENDSRC
```

macro `\SRCcontent` is defined as

```
\SRConeline{second line}
\SRCblock{  }{anotherblock}
```

and into the file `\jobname.ds` we write

```
\DEFSRC {InternalLabel}{Formatted label}
\SRCbeginline :second line\SRCendline
\SRCreadblock {  }{anotherblock}
\ENDDEFSRC
```

Implementation

File \jobname.ds

This file is read in the second pass of TeX. The macros are defined as

```
\def\DEFSRC#1#2{\def\SRCid{#1}
  \sdef{SRCtit:#1}{#2}
  \sdef{SRCcon:#1}{}}
\def\ADDSRC#1{\def\SRCid{#1}}
\def\SRCbeginline:{\bgroup\setSRC \catcode`\|=12 \SRCbeginlineA}
{\catcode`\|=0 \catcode`\\=12
  |gdef|SRCbeginlineA#1\SRCendline{|egroup
    |expandafter|addto|csname SRCcon:|SRCid|endcsname
      {|SRConeline{#1}}%
    |ignorespaces}}
\def\SRCreadblock{\begingroup\afterassignment\SRCreadblockA\catcode32=13}
\def\SRCreadblockA#1{\gdef\SRCods{#1}\endgroup\SRCreadblockB}
\def\SRCreadblockB#1{\expandafter\addto
  \csname SRCcon:\SRCid\expandafter\endcsname\expandafter{%
    \expandafter\SRCblock\expandafter{\SRCods}{#1}}}
\let\ENDDEFSRC\relax
```

Implementation
Macro `\SRCcon:xxx`

When reading file `\jobname.ds`, we have the definitions

```
\def\SRConeline#1{\immediate\write\SRCfile{\SRCodsazeni#1}}
\def\SRCblock#1#2{\begingroup
  \addto\SRCodsazeni{#1}%
  \csname SRCcon:#2\endcsname
  \endgroup}
```

So after expansion of

```
\DEFSRC {InternalLabel}{Formatted label}
\SRCbeginline :second line\SRCendline
\SRCreadblock {   }{anotherblock}
\ENDDEFSRC
```

macro `\SRCcon:InternalLabel` is defined as

```
\SRConeline{second line}\SRCblock{   }{anotherblock}
```

and this recursively expands to `\SRConeline`’s and `\write`’s.

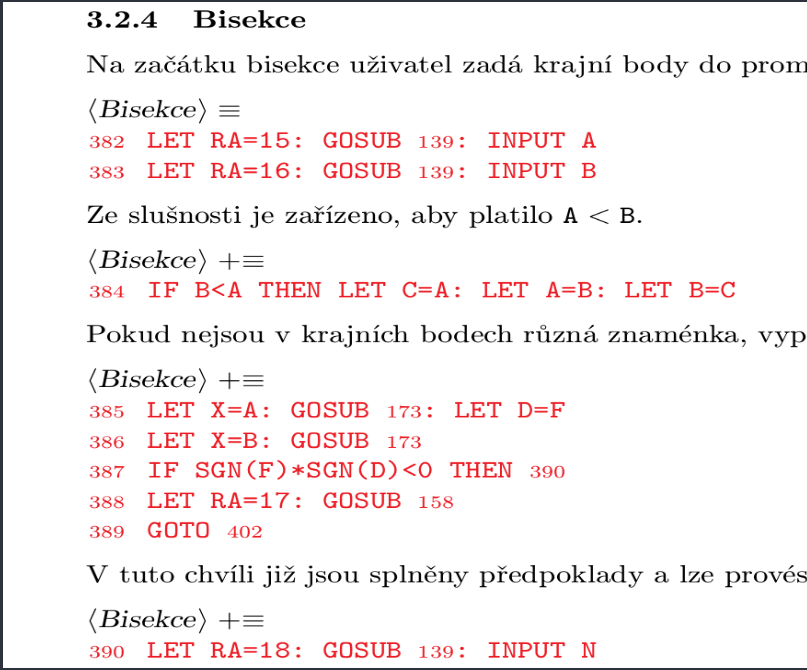
Implementation
Hooks

```
\SRChook

\endSRChook

\SRCdshook

\SRClinehook

\SRCwritehook
```

Applications
Cross-references

Hooks can be useful when generating programs in languages which use line numbers.

```
1190 let ra=15: gosub 2080: input a
1200 let ra=16: gosub 2080: input b
1210 if b<a then let c=a: let a=b: let b=c
1220 let x=a: gosub 3100: let d=f
1230 let x=b: gosub 3100
1240 if sgn(f)*sgn(d)<0 then 1270
1250 let ra=17: gosub 2270
1260 goto 1390
1270 let ra=18: gosub 2080: input n
1280 for i=1 to n
1290 let c=(a+b)/2
1300 let x=a: gosub 3100: let d=f
1310 let x=c: gosub 3100
1320 if sgn(f)*sgn(d)<0 then 1340
1330 let a=c: goto 1350
1340 let b=c
1350 print i;":";a,b
1360 next i
1370 let c=(a+b)/2
1380 print bz$;c;bn$
1390 goto 1970
```



3.2.4 Bisekce

Na začátku bisekce uživatel zadá krajní body do prom

⟨*Bisekce*⟩ ≡

```
382 LET RA=15: GOSUB 139: INPUT A
383 LET RA=16: GOSUB 139: INPUT B
```

Ze slušnosti je zařízeno, aby platilo `A` < `B`.

⟨*Bisekce*⟩ +≡

```
384 IF B<A THEN LET C=A: LET A=B: LET B=C
```

Pokud nejsou v krajních bodech různá znaménka, vyp

⟨*Bisekce*⟩ +≡

```
385 LET X=A: GOSUB 173: LET D=F
386 LET X=B: GOSUB 173
387 IF SGN(F)*SGN(D)<0 THEN 390
388 LET RA=17: GOSUB 158
389 GOTO 402
```

V tuto chvíli již jsou splněny předpoklady a lze provés

⟨*Bisekce*⟩ +≡

```
390 LET RA=18: GOSUB 139: INPUT N
```

Applications
Imitation of subprograms

First we define a “function” and reserve identifiers for its arguments and output value.

```
\BEGSRC<SDR>{Concat solutions}
SDRout=SDRin2.clone();
if (SDRin1.max > SDRin2.max) {SDRout.max = SDRin1.max;}
else {SDRout.max = SDRin2.max;}
SDRout.position = SDRin1.position.concat(SDRin2.position);}
\ENDSRC
```

Then we set values of the identifiers and “call” the “function”.

```
\BEGSRC
SDRin1=mainsolution.clone();
SDRin1.position=mainsolution.position.clone();
SDRin2=ZPRout.clone();
SDRin2.position=ZPRout.position.clone();
|<SDR>
mainsolution=SDRout.clone();
mainsolution.position=SDRout.position.clone();
\ENDSRC
```

Applications
Multilingual output

TEX macros can define a metalanguage and generate the source code in several programming languages simultaneously.

```
\def\variant#1#2#3{\ifOR \sdef{#1}##1{#2}\else \sdef{#1}##1{#3}\fi}
\variant{integer}{number(38)}{decimal(38)}
{\catcode'\_=12 \globaldefs=1
  \variant{rownum}{rownum}{(row_number() over (order by #1))}}

{\catcode'\@=13 \gdef@#1@#2@{\csname#1\endcsname{#2}}}
\sdef{}#1{@}
\def\SRChook{\catcode'\@=13 }

\BEGSRC
select cast(NN/p as @integer@@), st+1
from FindN1,
    (select @rownum@p@ r, p from SmallFactors) a
where st=r
\ENDSRC
```

Final words

Final words

Thanks for attention

and

happy TeXing!