1 The Bulgarian language

The file `bulgaria.dtx`[1] is derived from the files `greek.dtx`, `frenchb.dtx` and `russianb.dtx`[2]. It defines all the language definition macros for the Bulgarian language.

This version of the file has been designed to be used with LaTeX only[3]. If you are still using LaTeX-2.09, you *should* consider switching to LaTeX 2$_\varepsilon$!

The command `\selectlanguage{bulgarian}`, switches to the Bulgarian language with the following effects:

1. Bulgarian hyphenation patterns are made active;
2. `\today` prints the date in Bulgarian;
3. the caption names are translated into Bulgarian;
4. the default items in itemize environment are set to '—' instead of •;
5. the default items in enumerate environment are changed;
6. the commands for adjusting the labels of enumerate environment are provided, see table 3;
7. the commands `\Azbuk` and `\azbuk` are provided, working like `\Alph` and `\alph` but producing Cyrillic letters;
8. the vertical spacing in the general LaTeX lists is shortened, a hook to reset standard LaTeX settings is provided (`\BulgarianListSpacingfalse`) ;
9. the first paragraph of each section is indented;
10. after the number of sections, tables and figures a dot is inserted;
11. the character `"` is made active. In table 1 an overview is given of its purpose;
12. the command `\No` is a shortcut for typing the Cyrillic nomerosign;
13. a command `\degre` is provided to typeset temperatures (e.g., "`20\,\degres C`" with a thin space), or for alcohols' strengths (e.g., "`45\degres`" with *no* space in Bulgarian);
14. commands for some Bulgarian math functions are defined: `\sh`, `\ch`, `\tg`, `\arctg`, `\arccotg`, `\th`, `\cotg`, `\cosec`;
15. commands for Cyrillic in math-mode are provided (note: they require the package `mathtext`);
16. commands for some Bulgarian math Cyrillic symbols are defined: `\Prob` (probability), `\Expectation`, `\Variance`, `\nod` and `\NOD`, `\nok` and `\NOK` and `\Proj` (projection);

---

[1] The file described in this section has version number v1.0 and was last revised on 2000/08/02.

[2] Prof. Tinko Tinchev provided information about the Bulgarian typesetting traditions and prof. Dimiter Skordev gave some suggestions to improve the code.

[3] In some next version may be there will be support for PlainTeX users.

17. a command `\nombre` is provided to ease the typesetting of numbers: it works both in text and in math-mode: inputting `\nombre{3141,592653}` will format this number properly according to the current language (Bulgarian or non-Bulgarian) [4]. The command `\nombre` is a contribution of Vincent Jalby using ideas of David Carlisle in comma.sty.

`\cyrillictext`
`\latintext`

The commands `\cyrillictext` and `\latintext` can be used to switch to Cyrillic or Latin fonts. These are declarations.

`\textcyrillic`
`\textlatin`

The commands `\textcyrillic` and `\textlatin` both take one argument which is then typeset using the requested font encoding.

The user may choose between different available Cyrillic font encodings—`X2`, `T2A`, `T2B`, `T2C`, `LCY`, or `LWN`. If the user wants to use another font encoding than the default (`T2A`), he has to load the corresponding file *before* `bulgaria.sty`. This may be done in the following way:

```
% override the default T2C encoding used in Babel
\usepackage[LCY,OT1]{fontenc}
\usepackage[english,bulgarian]{babel}
```

Note: for the Bulgarian language, it is preferable to use encodings `T2A`, `T2B` or `T2C`, because other encodings does not contain Latin letters, and users should be very careful to switch the language every time they want to typeset a Latin word inside a Bulgarian phrase or vice versa.

For Bulgarian language the character `"` is made active. In table 1 an overview is given of its purpose.

`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`"---`	Cyrillic emdash in plain text.
`"--*`	Cyrillic emdash for denoting direct speech.
`""`	like `"-`, but producing no hyphen sign (for compound words with hyphen, e.g. `x-""y` or some other signs as “disable/enable”).
`"~`	for a compound word mark without a breakpoint.
`"=`	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
`",`	thin space for initials with a breakpoint in following surname.
`` "` ``	for German left double quotes (looks like „).
`"'`	for German right double quotes (looks like “).
`"<`	for French left double quotes (looks like $\ll$).
`">`	for French right double quotes (looks like $\gg$).

Table 1: The extra definitions made by `bulgaria`

The quotes in table 1 can also be typeset by using the commands in table 2.

---

[4] In math-mode the comma has to be surrounded with braces to avoid a spurious space being inserted after it (see the TEXbook p. 134). Besides this, each slice of three digits should be separated either with a comma in English or with a space in Bulgarian.

`\cdash---`	Cyrillic emdash in plain text.
`\cdash--*`	Cyrillic emdash for denoting direct speech.
`\glqq`	for German left double quotes (looks like „).
`\grqq`	for German right double quotes (looks like “).
`\flqq`	for French left double quotes (looks like ≪).
`\frqq`	for French right double quotes (looks like ≫).
`\dq`	the original quotes character (").

Table 2: More commands which produce quotes, defined by babel

The French quotes are also available as ligatures ‘`<<`’ and ‘`>>`’ in 8-bit Cyrillic font encodings (`LCY`, `X2`, `T2*`) and as ‘`<`’ and ‘`>`’ characters in 7-bit Cyrillic font encodings (`OT2` and `LWN`).

`\enumazbuk`
`\enumarabic`
`\enumAzbuk`
`\enumRoman`
`\enumstandard`

Use these commands for adjusting the labels of enumerate lists. After each of them the labels will be set according to the table 3. Of course when you change the language, the default values of the labels will be restored. By default the command `\enumarabic` is active. The command `\enumstandard` restores the standard labels of lists.

Command	first level	second level	third level	fourth level
`\enumazbuk`	small Cyrillic letters	undefined	undefined	undefined
`\enumarabic`	Arabic numbers	small Cyrillic letters	undefined	undefined
`\enumAzbuk`	capital Cyrillic letters	Arabic numbers	small Cyrillic letters	undefined
`\enumRoman`	big Roman numbers	capital Cyrillic letters	Arabic numbers	small Cyrillic letters
`\enumstandard`	arabic numbers	small Latin letters	small Roman numbers	capital Latin letters

Table 3: Commands for adjusting the labels of the Bulgarian enumerate lists

Some things `bulgaria` should do, but doesn't:

- In Bulgarian mathematical fractions the text style is never used. Use the display style instead: use $(a+1)/b$ or $\dfrac{a+1}{b}$ instead of $\frac{a+1}{b}$.
- Bulgarian large operators should be typed with limits. Use $\int\limits_{-\infty}^{+\infty}$ and $\sum\limits_{n=1}^{n}$ instead of $\int_{-\infty}^{+\infty}$ and $\sum_{n=1}^{n}$.

1.1 Preliminaries

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
⟨*code⟩
\LdfInit{bulgarian}{captionsbulgarian}
```

When this file is read as an option, i.e. by the `\usepackage` command, `bulgaria` could be an 'unknown' language in which case we have to make it known. So we check for the existence of `\l@bulgarian` to see whether we have to do something here.

```
\ifx\l@bulgarian\@undefined
  \@nopatterns{Bulgarian}
  \adddialect\l@bulgarian0\fi
```

`\bulgarianhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
\def\bulgarianhyphenmins{\tw@\tw@}
```

To check the format in use (plain or LaTeX), we'll need macros to hold the names of the plain and LaTeX 2$_\varepsilon$ formats.

```
\def\PlainFmtName{plain}
\def\LaTeXeFmtName{LaTeX2e}
```

`\if@Two@E` We will need a new 'if' : `\if@Two@E` is true if and only if LaTeX 2$_\varepsilon$ is running *not* in compatibility mode. It is used in the definitions of the command `\nombre`. The definition is somewhat complicated, due to the fact that `\if@compatibility` is not recognised as a `\if` in LaTeX-2.09 based formats.

```
\newif\if@Two@E \@Two@Etrue
\def\@FI@{\fi}
\ifx\@compatibilitytrue\@undefined
  \@Two@Efalse \def\@FI@{\relax}
\else
  \if@compatibility \@Two@Efalse \fi
\@FI@
```

It is best to use LaTeX 2$_\varepsilon$'s font changing commands, and to emulated those we need when they are not available, as in PlainTeX or LaTeX-2.09. Be aware that old commands `\sc`, `\it`, *etc.* exist in LaTeX 2$_\varepsilon$, but they behave like they did in LaTeX-2.09 (i. e., they switch back to `\normalfont` instead of keeping the other font attributes unchanged).

```
\ifx\emph\@undefined
  \ifx\em\@undefined
     \let\emph\relax
  \else
     \def\emph#1{\em #1}
  \fi
\fi
```

1.2 Encoding issues

`\cyrillicencoding` We parse the `\cdp@list` containing the encodings known to LaTeX in the order they were loaded. We set the `\cyrillicencoding` to the *last* loaded encoding in the list of supported Cyrillic encodings: `OT2`, `LWN`, `LCY`, `X2`, `T2C`, `T2B`, `T2A`, if any.

```
\def\reserved@a#1#2{%
   \edef\reserved@b{#1}%
   \edef\reserved@c{#2}%
   \ifx\reserved@b\reserved@c
     \let\cyrillicencoding\reserved@c
   \fi}
\def\cdp@elt#1#2#3#4{%
   \reserved@a{#1}{OT2}%
   \reserved@a{#1}{LWN}%
   \reserved@a{#1}{LCY}%
   \reserved@a{#1}{X2}%
   \reserved@a{#1}{T2C}%
   \reserved@a{#1}{T2B}%
   \reserved@a{#1}{T2A}}
\cdp@list
```

Now, if `\cyrillicencoding` is undefined, then the user did not load any of supported encodings. So, we have to set `\cyrillicencoding` to some default value. We test the presence of the encoding definition files in the order from less preferable to more preferable encodings. We use the lowercase names (i.e., `lcyenc.def` instead of `LCYenc.def`).

```
\ifx\cyrillicencoding\undefined
  \IfFileExists{ot2enc.def}{\def\cyrillicencoding{OT2}}\relax
  \IfFileExists{lwnenc.def}{\def\cyrillicencoding{LWN}}\relax
  \IfFileExists{lcyenc.def}{\def\cyrillicencoding{LCY}}\relax
  \IfFileExists{x2enc.def}{\def\cyrillicencoding{X2}}\relax
  \IfFileExists{t2cenc.def}{\def\cyrillicencoding{T2C}}\relax
  \IfFileExists{t2benc.def}{\def\cyrillicencoding{T2B}}\relax
  \IfFileExists{t2aenc.def}{\def\cyrillicencoding{T2A}}\relax
```

If `\cyrillicencoding` is still undefined, then the user seems not to have a properly installed distribution. A fatal error.

```
  \ifx\cyrillicencoding\undefined
    \PackageError{babel}%
      {No Cyrillic encoding definition files were found}%
      {Your installation is incomplete.\MessageBreak
       You need at least one of the following files:\MessageBreak
       \space\space
       x2enc.def, t2aenc.def, t2benc.def, t2cenc.def,\MessageBreak
       \space\space
       lcyenc.def, lwnenc.def, ot2enc.def.}%
  \else
```

We avoid `\usepackage[\cyrillicencoding]{fontenc}` because we don't want to force the switch of `\encodingdefault`.

```
    \lowercase
       \expandafter{\expandafter\input\cyrillicencoding enc.def\relax}%
  \fi
\fi
```

Now we define two commands that offer the possibility to switch between Cyrillic and Roman encodings.

`\cyrillictext`

The command `\cyrillictext` will switch from Latin font encoding to the Cyrillic font encoding. This assumes that the 'normal' font encoding is a Latin one. This command is a *declaration*, for shorter pieces of text the command `\textcyrillic` should be used.

```
\DeclareRobustCommand{\cyrillictext}{%
  \fontencoding\cyrillicencoding\selectfont
  \let\encodingdefault\cyrillicencoding}
```

`\textcyrillic`

This command takes an argument which is then typeset using the requested font encoding. In order to avoid many encoding switches it operates in a local scope.

```
\DeclareRobustCommand{\textcyrillic}[1]{{\cyrillictext #1}}
```

`\extrasbulgarian`
`\noextrasbulgarian`

The macro `\extrasbulgarian` will perform all the extra definitions needed for the Bulgarian language. The macro `\noextrasbulgarian` is used to cancel the actions of `\extrasbulgarian`.

The first action we define is to switch on the selected Cyrillic encoding whenever we enter Bulgarian language.

```
\addto\extrasbulgarian{\cyrillictext}
```

When the encoding definition file was processed by LaTeX the current font encoding is stored in `\latinencoding`, assuming that LaTeX uses `T1` or `OT1` as default. Therefore we switch back to `\latinencoding` whenever the Bulgarian language is no longer 'active'.

```
\addto\noextrasbulgarian{\latintext}
```

Since the `X2` encoding does not contain Latin letters, we should make some redefinitions of LaTeX macros which implicitly produce Latin letters.

```
\expandafter\ifx\csname T@X2\endcsname\relax\else
```

We put `\latinencoding` in braces to avoid problems with `\@alph` inside minipages (e.g., footnotes inside minipages) where `\@alph` is expanded and we get for example '`\fontencoding OT1`' (`\fontencoding` is robust).

```
  \def\@alph#1{{\fontencoding{\latinencoding}\selectfont
    \ifcase#1\or
      a\or b\or c\or d\or e\or f\or g\or h\or
      i\or j\or k\or l\or m\or n\or o\or p\or
      q\or r\or s\or t\or u\or v\or w\or x\or
      y\or z\else\@ctrerr\fi}}%
  \def\@Alph#1{{\fontencoding{\latinencoding}\selectfont
    \ifcase#1\or
      A\or B\or C\or D\or E\or F\or G\or H\or
      I\or J\or K\or L\or M\or N\or O\or P\or
      Q\or R\or S\or T\or U\or V\or W\or X\or
      Y\or Z\else\@ctrerr\fi}}%
```

Unfortunately, the commands `\AA` and `\aa` are not encoding dependent in LaTeX (unlike e.g., `\oe` or `\DH`). They are defined as `\r{A}` and `\r{a}`. This leads to unpredictable results when the font encoding does not contain the Latin letters 'A' and 'a' (like `X2`).

```
  \DeclareTextSymbolDefault{\AA}{OT1}
  \DeclareTextSymbolDefault{\aa}{OT1}
  \DeclareTextCommand{\aa}{OT1}{\r a}
  \DeclareTextCommand{\AA}{OT1}{\r A}
\fi
```

1.3 Caption Names and Date

In Bulgarian, captions in figures and tables should be printed with '.' instead the standard ':', so we add a hook called `\CaptionSeparator` to the definition of `\@makecaption` (this definition set in classes.dtx is frozen for LaTeX 2$_{\varepsilon}$ according to Frank Mittelbach).

```
\def\CaptionSeparator{\string:\space}
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{#1\CaptionSeparator #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1\CaptionSeparator #2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
\addto\extrasbulgarian{%
  \def\CaptionSeparator{\string.\space}}
\addto\noextrasbulgarian{%
  \def\CaptionSeparator{\string:\space}}
```

The next step consists of defining the Bulgarian equivalents for the LaTeX caption names.

`\captionsbulgarian`

The macro `\captionsbulgarian` defines all strings used in the four standard document classes provided with LaTeX. If you do not like some of these names, it is easy to change them in the preamble *after* loading `bulgaria` (or in your file `bulgaria.cfg`), e.g `\addto\captionsbulgarian{\def\tablename{Tab.}}` will print the word 'Tablica' abbreviated.

```
\ifx\fmtname\PlainFmtName
\else
\addto\captionsbulgarian{%
  \def\prefacename{%
%   {\cyr\CYRU\cyrv\cyro\cyrd}}%
    {\CYRP\cyrr\cyre\cyrd\cyrg\cyro\cyrv\cyro\cyrr}}%
  \def\refname{%
    {\CYRL\cyri\cyrt\cyre\cyrr\cyra\cyrt\cyru\cyrr\cyra}}%
  \def\abstractname{%
    {\CYRR\cyre\cyrz\cyryu\cyrm\cyre}}%
  \def\bibname{%
%   {\cyr\cyrK\cyrn\cyri\cyrzh\cyrn\cyri\cyrn\cyra}}%
    {\CYRB\cyri\cyrb\cyrl\cyri\cyro\cyrg\cyrr\cyra\cyrf\cyri\cyrya}}%
  \def\chaptername{{\CYRG\cyrl\cyra\cyrv\cyra}}%
  \def\appendixname{%
    {\CYRP\cyrr\cyri\cyrl\cyro\cyrzh\cyre\cyrn\cyri\cyre}}%
  \def\contentsname{%
```

```
    {\CYRS\cyrhrdsn\cyrd\cyrhrdsn\cyrr\cyrzh\cyra\cyrn\cyri\cyre}}%
  \def\listfigurename{%
    {\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\space\cyrn\cyra\space
      \cyrf\cyri\cyrg\cyru\cyrr\cyri\cyrt\cyre}}%
  \def\listtablename{%
    {\CYRS\cyrp\cyri\cyrs\cyrhrdsn\cyrk\space\cyrn\cyra\space
      \cyrt\cyra\cyrb\cyrl\cyri\cyrc\cyri\cyrt\cyre}}%
  \def\indexname{%
%    {\CYRA\cyrz\cyrb\cyru\cyrch\cyre\cyrn
%      \space\cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl}}%
    {\CYRP\cyrr\cyre\cyrd\cyrm\cyre\cyrt\cyre\cyrn
      \space\cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl}}%
  \def\authorname{%
    {\CYRI\cyrm\cyre\cyrn\cyre\cyrn\space
      \cyru\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl}}%
  \def\figurename{{\CYRF\cyri\cyrg.}}%
  \def\tablename{{\CYRT\cyra\cyrb\cyrl\cyri\cyrc\cyra}}%
  \def\partname{%
%   {\CYRD\cyrya\cyrl}}%
    {\CYRCH\cyra\cyrs\cyrt}}%
  \def\enclname{{\CYRP\cyrr\cyri\cyrl\cyro\cyrzh.}}%
  \def\ccname{{\CYRK\cyro\cyrp\cyri\cyre\space\cyrd\cyro}}%
  \def\headtoname{{\CYRD\cyro}}%
  \def\pagename{{\cyrs\cyrt\cyrr.}}%
  \def\seename{{\cyrv\cyrzh.}}%
  \def\alsoname{{\cyrv\cyrzh.\space\cyrs\cyrhrdsn\cyrshch\cyrt\cyro}}%
  \def\proofname{{\CYRD\cyro\cyrk\cyra\cyrz\cyra\cyrt\cyre\cyrl
      \cyrs\cyrt\cyrv\cyro}}%
  }
\fi
```

`\datebulgarian`

The macro `\datebulgarian` redefines the command `\today` to produce Bulgarian dates.

```
\def\datebulgarian{%
  \def\today{\number\day~\ifcase\month\or
    \cyrya\cyrn\cyru\cyra\cyrr\cyri\or
    \cyrf\cyre\cyrv\cyrr\cyru\cyra\cyrr\cyri\or
    \cyrm\cyra\cyrr\cyrt\or
    \cyra\cyrp\cyrr\cyri\cyrl\or
    \cyrm\cyra\cyrishrt\or
    \cyryu\cyrn\cyri\or
    \cyryu\cyrl\cyri\or
    \cyra\cyrv\cyrg\cyru\cyrs\cyrt\or
    \cyrs\cyre\cyrp\cyrt\cyre\cyrm\cyrv\cyrr\cyri\or
    \cyro\cyrk\cyrt\cyro\cyrm\cyrv\cyrr\cyri\or
    \cyrn\cyro\cyre\cyrm\cyrv\cyrr\cyri\or
    \cyrd\cyre\cyrk\cyre\cyrm\cyrv\cyrr\cyri\fi
    \space\number\year~\cyrg.}}
```

1.4 Punctuation

One more thing `\extrasbulgarian` needs to do is to make sure that `\frenchspacing` is in effect. The execution of `\noextrasbulgarian` will switch it off again.

```
\addto\extrasbulgarian{\bbl@frenchspacing}
\addto\noextrasbulgarian{\bbl@nonfrenchspacing}
```

We specify that the Bulgarian group of shorthands should be used.

```
\addto\extrasbulgarian{\languageshorthands{bulgarian}}
```

The category code of the character '`"`' is made `\active`. It is used as indicated in table 1.

```
\initiate@active@char{"}
\addto\extrasbulgarian{%
  \bbl@activate{"}}
\addto\noextrasbulgarian{%
  \bbl@deactivate{"}}
```

To be able to define the function of this character, we first define a couple of 'support' macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as '`"`'.

```
\begingroup \catcode`\"12
\def\reserved@a{\endgroup
  \def\@SS{\mathchar"7019 }
  \def\dq{"}}
\reserved@a
```

Now we can define the double quote macros: German and French quotes. We use definitions of these quotes made in babel.sty. The French quotes are contained in the `T2*` encodings.

```
\declare@shorthand{bulgarian}{"`}{\glqq}
\declare@shorthand{bulgarian}{"'}{\grqq}
\declare@shorthand{bulgarian}{"<}{\flqq}
\declare@shorthand{bulgarian}{">}{\frqq}
```

Some additional commands:

```
\declare@shorthand{bulgarian}{""}{\hskip\z@skip}
\declare@shorthand{bulgarian}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
\declare@shorthand{bulgarian}{"=}{\nobreak-\hskip\z@skip}
\declare@shorthand{bulgarian}{"|}{%
  \textormath{\nobreak\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
```

The next two macros for `"-` and `"---` are somewhat different. We must check whether the second token is a hyphen character:

```
\declare@shorthand{bulgarian}{"-}{%
```

If the next token is '`-`', we typeset an emdash, otherwise a hyphen sign:

```
  \def\bulgarian@sh@tmp{%
    \if\bulgarian@sh@next-\expandafter\bulgarian@sh@emdash
    \else\expandafter\bulgarian@sh@hyphen\fi
  }%
```

TeX looks for the next token after the first '`-`': the meaning of this token is written to `\bulgarian@sh@next` and `\bulgarian@sh@tmp` is called.

```
  \futurelet\bulgarian@sh@next\bulgarian@sh@tmp}
```

Here are the definitions of hyphen and emdash. First the hyphen:

```
\def\bulgarian@sh@hyphen{%
  \nobreak\-\bbl@allowhyphens}
```

For the emdash definition, there are the two parameters: we must 'eat' two last hyphen signs of our emdash. . . :

```
\def\bulgarian@sh@emdash#1#2{\cdash-#1#2}
```

`\cdash` . . . these two parameters are useful for another macro: `\cdash`:

```
%\ifx\cdash\undefined % should be defined earlier
\def\cdash#1#2#3{\def\tempx@{#3}%
\def\tempa@{-}\def\tempb@{~}\def\tempc@{*}%
 \ifx\tempx@\tempa@\@Acdash\else
  \ifx\tempx@\tempb@\@Bcdash\else
   \ifx\tempx@\tempc@\@Ccdash\else
    \errmessage{Wrong usage of cdash}\fi\fi\fi}
```

second parameter (or third for `\cdash`) shows what kind of emdash to create in next step

`"---` ordinary (plain) Cyrillic emdash inside text: an unbreakable thin space will be inserted before only in case of a *space* before the dash (it is necessary for dashes after display maths formulae: there could be lists, enumerations etc. started with "— where a is ..." i.e., the dash starts a line). (Firstly there were planned rather soft rules for user: he may put a space before the dash or not. But it is difficult to place this thin space automatically, i.e., by checking modes because after display formulae TEX uses horizontal mode. Maybe there is a misunderstanding? Maybe there is another way?) After a dash a breakable thin space is always placed;

```
% What is more grammatically: .2em or .2\fontdimen6\font ?
\def\@Acdash{\ifdim\lastskip>\z@\unskip\nobreak\hskip.2em\fi
  \cyrdash\hskip.2em\ignorespaces}%
```

`"--~` emdash in compound names or surnames (like Mendeleev–Klapeiron); this dash has no space characters around; after the dash some space is added `\exhyphenalty`.
NOTE: This is not used in Bulgarian, but is preserved here for compatibility with the macro `\cdash` in `russianb` and `ukraineb`.

```
\def\@Bcdash{\leavevmode\ifdim\lastskip>\z@\unskip\fi
  \nobreak\cyrdash\penalty\exhyphenpenalty\hskip\z@skip\ignorespaces}%
```

`"--*` for denoting direct speech (a space like `\enskip` must follow the emdash);

```
\def\@Ccdash{\leavevmode
  \nobreak\cyrdash\nobreak\hskip.35em\ignorespaces}%
%\fi
```

`\cyrdash` Finally the macro for "body" of the Cyrillic emdash. The `\cyrdash` macro will be defined in case this macro hasn't been defined in a fontenc file. For T2* fonts, cyrdash will be placed in the code of the English emdash thus it uses ligature `---`.

```
% Is there an IF necessary?
\ifx\cyrdash\undefined
  \def\cyrdash{\hbox to.8em{--\hss--}}
\fi
```

Here is another macro—to place thin space between initials. This macro used instead of `\,` allows hyphenation in the following surname.

```
\declare@shorthand{bulgarian}{",}{\nobreak\hskip.2em\ignorespaces}
```

`\mdqon` `\mdqoff` All that's left to do now is to define commands for switching on and of double quote activeness.

```
\def\mdqon{\bbl@activate{"}}
\def\mdqoff{\bbl@deactivate{"}}
```

1.5 Lists

We start with adding new enumeration styles for Bulgarian manuscripts—with Cyrillic letters

`\Azbuk` We begin by defining `\Azbuk` which works like `\Alph`, but produces (uppercase) Cyrillic letters instead of Latin ones. No Bulgarian letters are skipped.

```
\def\Azbuk#1{\expandafter\@Azbuk\csname c@#1\endcsname}
\def\@Azbuk#1{{\fontencoding{\cyrillicencoding}\selectfont
  \ifcase#1\or
  \CYRA\or\CYRB\or\CYRV\or\CYRG\or\CYRD\or\CYRE\or\CYRZH\or
  \CYRZ\or\CYRI\or\CYRISHRT\or\CYRK\or\CYRL\or\CYRM\or\CYRN\or\CYRO\or
  \CYRP\or\CYRR\or\CYRS\or\CYRT\or\CYRU\or\CYRF\or\CYRH\or
  \CYRC\or\CYRCH\or\CYRSH\or\CYRSHCH\or\CYRHRDSN\or\CYRSFTSN\or\CYRYU\or
  \CYRYA\else\@ctrerr\fi}}
```

`\azbuk` The macro `\azbuk` is similar to `\alph`; it produces lowercase Bulgarian letters.

```
\def\azbuk#1{\expandafter\@azbuk\csname c@#1\endcsname}
\def\@azbuk#1{{\fontencoding{\cyrillicencoding}\selectfont
  \ifcase#1\or
  \cyra\or\cyrb\or\cyrv\or\cyrg\or\cyrd\or\cyre\or\cyrzh\or
  \cyrz\or\cyri\or\cyrishrt\or\cyrk\or\cyrl\or\cyrm\or\cyrn\or\cyro\or
  \cyrp\or\cyrr\or\cyrs\or\cyrt\or\cyru\or\cyrf\or\cyrh\or
  \cyrc\or\cyrch\or\cyrsh\or\cyrshch\or\cyrhrdsn\or\cyrsftsn\or\cyryu\or
  \cyrya\else\@ctrerr\fi}}
```

Now we make the vertical spacing in general lists shorter than the defaults provided by LaTeX. Note that the easy way, just changing values of vertical spacing parameters when entering Bulgarian and restoring them to their defaults on exit would not work. We will redefine `\@trivlist` as `\list` and `trivlist` rely on `\@trivlist`, so that all lists have common settings. If standard LaTeX settings are preferred, it is easy to issue the command `\BulgarianListSpacingfalse` in the preamble or in `bulgaria.cfg`. Please note that changing the flag `\BulgarianListSpacing` will *not* take effect immediately, but next time language Bulgarian is switched on.

The amount of vertical space before and after a list is given by `\topsep` + `\parskip` (+ `\partopsep` if the list starts a new paragraph). IMHO, `\parskip` should be added *only* when the list starts a new paragraph, so I subtract `\parskip` from `\topsep` and add it back to `\partopsep`; this will normally make no difference because `\parskip`'s default value is 0pt, but will be noticeable when `\parskip` is *not* null.
Of course, this code is only for LaTeX.

```
\newif\ifBulgarianListSpacing \BulgarianListSpacingtrue
\ifx\fmtname\PlainFmtName
\else
  \let\@trivlistBG\@trivlist
  \addto\extrasbulgarian{%
    \ifBulgarianListSpacing
      \def\@trivlist{%
          \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
          \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%
          \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
          \setlength{\partopsep}{0.4ex plus 0.2ex minus 0.2ex}%
          \addtolength{\topsep}{-\parskip}%
          \addtolength{\partopsep}{\parskip}%
          \@trivlistBG}%
    \fi}
  \addto\noextrasbulgarian{\let\@trivlist\@trivlistBG}
\fi
```

`\enumazbuk`
`\enumarabic`
`\enumAzbuk`
`\enumRoman`
`\enumstandard`

The labels of the Bulgarian enumerate-lists should be changed. On outer levels should be used big Roman numbers and capital Cyrillic letters. On inner levels should be used Arabic numbers and small Cyrillic letters. We suppose that there are one or two levels, so by default, for the first level we use Arabic numbers and for the second level we use small Cyrillic letters. We provide also commands for changing the labels of the lists. Use `\enumazbuk` before one level enumerate-list if you want small Cyrillic letters as labels. Use `\enumarabic` before two levels enumerate-lists. Use `\enumAzbuk` before three levels enumerate-lists, and use `\enumRoman` before four levels enumerate-lists. Of course when you change the language, the default values of the labels will be restored.

The style of labels with small Cyrillic letters is a), b), v), etc. instead of (a), (b), (v), etc. We add the the left parent to `\theenumN`, not to `\labelenumN`, because it should be printed by `\ref` too.

The command `\enumstandard` restores the standard labels of lists.

```
\ifx\fmtname\PlainFmtName
\else
  \let\@tniBG\theenumi
  \let\@lniBG\labelenumi
  \let\@tniiBG\theenumii
  \let\@lniiBG\labelenumii
  \let\@tniiiBG\theenumiii
  \let\@lniiiBG\labelenumiii
  \let\@tnivBG\theenumiv
  \let\@lnivBG\labelenumiv
  \let\@pniiBG\p@enumii
  \let\@pniiiBG\p@enumiii
  \let\@pnivBG\p@enumiv
  \def\enumstandard{%
      \let\theenumi\@tniBG
      \let\labelenumi\@lniBG
      \let\theenumii\@tniiBG
      \let\labelenumii\@lniiBG
      \let\theenumiii\@tniiiBG
      \let\labelenumiii\@lniiiBG
      \let\theenumiv\@tnivBG
      \let\labelenumiv\@lnivBG
      \let\p@enumii\@pniiBG
      \let\p@enumiii\@pniiiBG
      \let\p@enumiv\@pnivBG}%
  \def\enumazbuk{%
      \def\theenumi{\azbuk{enumi})}%
      \def\labelenumi{\theenumi}}%
  \def\enumarabic{%
      \def\theenumi{\arabic{enumi}}%
      \def\labelenumi{\theenumi.}%
      \def\theenumii{\azbuk{enumii})}%
      \def\labelenumii{\theenumii}
      \def\p@enumii{\theenumi}}%
  \def\enumAzbuk{%
      \def\theenumi{\Azbuk{enumi}}%
      \def\labelenumi{\theenumi.}%
      \def\theenumii{\arabic{enumii}}%
      \def\labelenumii{\theenumii.}%
      \def\theenumiii{\azbuk{enumiii})}%
      \def\labelenumiii{\theenumiii}%
      \def\p@enumii{\theenumi.}%
      \def\p@enumiii{\theenumi.\theenumii}}%
  \def\enumRoman{%
      \def\theenumi{\Roman{enumi}}%
      \def\labelenumi{\theenumi.}%
      \def\theenumii{\Azbuk{enumii}}%
      \def\labelenumii{\theenumii.}%
      \def\theenumiii{\arabic{enumiii}}%
      \def\labelenumiii{\theenumiii.}%
      \def\theenumiv{\azbuk{enumiv})}%
      \def\labelenumiv{\theenumiv}%
      \def\p@enumii{\theenumi.}%
      \def\p@enumiii{\theenumi.\theenumii.}%
      \def\p@enumiv{\theenumi.\theenumii.\theenumiii}}%
  \addto\extrasbulgarian{\enumarabic}
  \addto\noextrasbulgarian{\enumstandard}
\fi
```

The • is never used in the Bulgarian itemize-lists, a long dash '—' is preferred for all levels.

```
\ifx\fmtname\PlainFmtName
\else
  \let\@ltiBG\labelitemi
  \let\@ltiiBG\labelitemii
  \let\@ltiiiBG\labelitemiii
  \let\@ltivBG\labelitemiv
  \addto\extrasbulgarian{%
      \def\labelitemi{\cyrdash}%
      \def\labelitemii{\cyrdash}%
      \def\labelitemiii{\cyrdash}%
      \def\labelitemiv{\cyrdash}}
  \addto\noextrasbulgarian{\let\labelitemi\@ltiBG
                           \let\labelitemii\@ltiiBG
                           \let\labelitemiii\@ltiiiBG
                           \let\labelitemiv\@ltivBG}
\fi
```

1.6 Sections

In Bulgarian the first paragraph of each section should be indented, this is another difference with US-English. Add this code only in LaTeX.

```
\ifx\fmtname\PlainFmtName
\else
 \let\@aifBG\@afterindentfalse
 \addto\extrasbulgarian{\let\@afterindentfalse\@afterindenttrue
                         \@afterindenttrue}
 \addto\noextrasbulgarian{\let\@afterindentfalse\@aifBG
                           \@afterindentfalse}
\fi
```

After the number of the sections there should be a dot and the space should be regular. We do not add this dot to the table of contents because this would make it inconsistent in multi-lingual documents.

```
\ifx\fmtname\PlainFmtName
\else
 \let\@scfBG\@seccntformat
 \addto\extrasbulgarian{%
     \def\@seccntformat#1{\csname the#1\endcsname.\space}}
 \addto\noextrasbulgarian{\let\@seccntformat\@scfBG}
\fi
```

1.7 Formatting numbers

This section is borrowed from `french.dtx`.

In English the decimal part starts with a point and thousands should be separated by a comma: an approximation of 1000π should be inputed as `$3{,}141.592{,}653$` in math-mode and as `3,141.592,653` in text.
In Bulgarian the decimal part starts with a comma and thousands should be separated by a space; the same approximation of 1000π should be inputed as `$3\;141{,}592\;653$` in math-mode and as something like `3~141,592~653` in text. Remember braces are mandatory around the comma in math-mode, the reason is mentioned in the TeXbook p. 134: the comma is of type `\mathpunct` (thus normally followed by a space) while the point is of type `\mathord` (no space added).

Thierry Bouche suggested that a second type of comma, of type `\mathord` would be useful in math-mode, and proposed to introduce a command (named `\decimalsep` in this package), the expansion of which would depend on the current language.

Vincent Jalby suggested a command `\nombre` to conveniently typeset numbers: inputting `\nombre{3141,592653}` either in text or in math-mode will format this number properly according to the current language (Bulgarian or other).

`\nombre` accepts an optional argument which happens to be useful with the package 'dcolumn', it specifies the decimal separator used in the *source code*:

```
\newcolumntype{d}{D{,}{\decimalsep}{-1}}
\begin{tabular}{d}\hline
```

```
  3,14 \\
  \nombre[,]{123,4567} \\
  \nombre[,]{9876,543}\\\hline
\end{tabular}
```

will print a column of numbers aligned on the decimal point (comma or point depending on the current language), each slice of 3 digits being separated by a space or a comma according to the current language.

`\decimalsep`
`\thousandsep`

We need a internal definition, valid in both text and math-mode, for the comma (`\@comma@`) and another one for the unbreakable fixed length space (no glue) used in Bulgarian (`\f@thousandsep`).

The commands `\decimalsep` and `\thousandsep` get default definitions (for the English language) when `bulgaria` is loaded; these definitions will be updated when the current language is switched to or from Bulgarian.

```
%    \begin{macrocode}
\mathchardef\m@comma="013B
\def\@comma@{\ifmmode\m@comma\else,\fi}
\def\f@thousandsep{\ifmmode\mskip5.5mu\else\penalty\@M\kern.3em\fi}
\newcommand{\decimalsep}{.}
\newcommand{\thousandsep}{\@comma@}
\addto\extrasbulgarian{%
            \def\decimalsep{\@comma@}%
            \def\thousandsep{\f@thousandsep}}
\addto\noextrasbulgarian{%
            \def\decimalsep{.}%
            \def\thousandsep{\@comma@}}
```

Signs can now be entered inside `\nombre`. When `\nombre` is used in text-mode, signs should be text symbols to get the series, shape... from the current text-font. When signs are not available in text-mode, we provide some defaults.

```
\providecommand{\textminus}{\textendash}%
\providecommand{\textplusminus}{\ensuremath{\pm}}
\providecommand{\textminusplus}{\ensuremath{\mp}}
\def\fb@minus{\ifmmode-\else\textminus\fi}
\def\fb@plusminus{\ifmmode\pm\else\textplusminus\fi}
\def\fb@minusplus{\ifmmode\mp\else\textminusplus\fi}
```

`\nombre`

The decimal separator used when *inputing* a number with `\nombre` *has to be a comma.* `\nombre` splits the inputed number into two parts: what comes before the first comma will be formatted by `\@integerpart` while the rest (if not empty) will be formatted by `\@decimalpart`. Both parts, once formatted separately will be merged together with between them, either the decimal separator `\decimalsep` or (in LaTeX 2_ε *only*) the optional argument of `\nombre`.

```
\if@Two@E
  \newcommand{\nombre}[2][\decimalsep]{\def\@decimalsep{#1}%
          \@@nombre#2\@empty,\@empty,\@nil}
\else
  \def\@decimalsep{\decimalsep}
  \newcommand{\nombre}[1]{\@nombre#1\@empty,\@empty,\@nil}
\fi
\def\@firstofmany#1#2,{#1}
\def\@@nombre#1,#2,#3\@nil{%
```

```
       \def\nb@sign{}%
       \edef\nb@first{\@firstofmany #1\@empty,}%
       \edef\nb@suite{\@secondoftwo #1\@empty,}%
       \if+\nb@first   \def\nb@sign{+}\fi
       \if-\nb@first   \def\nb@sign{\fb@minus}\fi
       \expandafter\ifx\nb@first\pm \def\nb@sign{\fb@plusminus}\fi
       \expandafter\ifx\nb@first\mp \def\nb@sign{\fb@minusplus}\fi
       \ifx\@empty\nb@sign
         \let\@tmp\nb@suite\edef\nb@suite{\nb@first\@tmp}%
       \fi
    \nb@sign\expandafter\@nombre\nb@suite#2,#3\@nil}
\def\@nombre#1,#2,#3\@nil{%
       \ifx\@empty#2%
         \@integerpart{#1}%
       \else
         \@integerpart{#1}\@decimalsep\@decimalpart{#2}%
       \fi}
```

The easiest bit is the decimal part: We attempt to read the first four digits of the decimal part, if it has less than 4 digits, we just have to print them, otherwise `\thousandsep` has to be appended after the third digit, and the algorithm is applied recursively to the rest of the decimal part.

```
\def\@decimalpart#1{\@@decimalpart#1\@empty\@empty\@empty}
\def\@@decimalpart#1#2#3#4{#1#2#3%
  \ifx\@empty#4%
  \else
    \thousandsep\expandafter\@@decimalpart\expandafter#4%
  \fi}
```

Formatting the integer part is more difficult because the slices of 3 digits start from the *bottom* while the number is read from the top! This (tricky) code is borrowed from David Carlisle's comma.sty.

```
\def\@integerpart#1{\@@integerpart{}#1\@empty\@empty\@empty}
\def\@@integerpart#1#2#3#4{%
  \ifx\@empty#2%
    \@addthousandsep#1\relax
  \else
    \ifx\@empty#3%
      \@addthousandsep\@empty\@empty#1#2\relax
    \else
      \ifx\@empty#4%
        \@addthousandsep\@empty#1#2#3\relax
      \else
        \@@integerpartafterfi{#1#2#3#4}%
      \fi
    \fi
  \fi}
\def\@@integerpartafterfi#1\fi\fi\fi{\fi\fi\fi\@@integerpart{#1}}
\def\@addthousandsep#1#2#3#4{#1#2#3%
  \if#4\relax
  \else
    \thousandsep\expandafter\@addthousandsep\expandafter#4%
  \fi}
```

1.8 Mathematics

Set up default Cyrillic math alphabets. To use Cyrillic letters in math mode user should load the `textmath` package *before* loading fontenc package (or babel). Note, that by default Cyrillic letters are taken from upright font in math mode (unlike Latin letters).

```
%\RequirePackage{textmath}
\@ifundefined{sym\cyrillicencoding letters}{}{%
\SetSymbolFont{\cyrillicencoding letters}{bold}\cyrillicencoding
  \rmdefault\bfdefault\updefault
\DeclareSymbolFontAlphabet\cyrmathrm{\cyrillicencoding letters}
```

And we need a few commands to be able to switch to different variants.

```
\DeclareMathAlphabet\cyrmathbf\cyrillicencoding
  \rmdefault\bfdefault\updefault
\DeclareMathAlphabet\cyrmathsf\cyrillicencoding
  \sfdefault\mddefault\updefault
\DeclareMathAlphabet\cyrmathit\cyrillicencoding
  \rmdefault\mddefault\itdefault
\DeclareMathAlphabet\cyrmathtt\cyrillicencoding
  \ttdefault\mddefault\updefault
%
\SetMathAlphabet\cyrmathsf{bold}\cyrillicencoding
  \sfdefault\bfdefault\updefault
\SetMathAlphabet\cyrmathit{bold}\cyrillicencoding
  \rmdefault\bfdefault\itdefault
}
```

Some math functions in Bulgarian math books have other names: e.g., `cotan` in Bulgarian is written as `cotg` etc. So we define a number of new math operators.

sinh

```
      \def\sh{\mathop{\operator@font sh}\nolimits}
```

cosh

```
      \def\ch{\mathop{\operator@font ch}\nolimits}
```

tan

```
      \def\tg{\mathop{\operator@font tg}\nolimits}
```

arctan

```
      \def\arctg{\mathop{\operator@font arctg}\nolimits}
```

arccotg

```
      \def\arccotg{\mathop{\operator@font arccotg}\nolimits}
```

tanh This macro conflicts with `\th` defined in Latin 1 encoding.

```
      \def\th{\mathop{\operator@font th}\nolimits}
```

cot

```
      \def\cotg{\mathop{\operator@font cotg}\nolimits}
```

csc

```
    \def\cosec{\mathop{\operator@font cosec}\nolimits}
```

And finally some other Bulgarian mathematical symbols:

```
\def\Prob{\mathop{\kern\z@\mathsf{P}}\nolimits}
\def\Expectation{\mathop{\kern\z@\mathsf{E}}\nolimits}
\def\Variance{\mathop{\kern\z@\mathsf{D}}\nolimits}
\def\nod{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrd.}}\nolimits}
\def\nok{\mathop{\cyrmathrm{\cyrn.\cyro.\cyrk.}}\nolimits}
\def\NOD{\mathop{\cyrmathrm{\CYRN\CYRO\CYRD}}\nolimits}
\def\NOK{\mathop{\cyrmathrm{\CYRN\CYRO\CYRK}}\nolimits}
\def\Proj{\mathop{\cyrmathrm{\CYRP\cyrr}}\nolimits}
```

1.9 Extra Utilities and Cleaning up

All that is left is to provide the Bulgarian user with some extra utilities.

`\No` A shortcut for the Cyrillic nomerosign:

```
\DeclareRobustCommand{\No}{%
  \ifmmode{\nfss@text{\textnumero}}\else\textnumero\fi}
```

`\degres` A macro for typesetting the abbreviation for 'degrees' (as in 'degrees Celsius'). As the bounding box of the character 'degree' has *very* different widths in CM/EC and PostScript fonts, we fix the width of the bounding box of `\degres` to 0.3 em, this lets the symbol 'degree' stick to the preceding (e.g., `45\degres`) or following character (e.g., `20\,\degres C`).

```
\DeclareRobustCommand*{\degres}{%
                         \leavevmode\hbox to 0.3em{\hss\degre\hss}}
```

Finally the macro-space used by some control sequences we do not need any longer, is freed.

```
\let\PlainFmtName\relax
\let\LaTeXeFmtName\relax
```

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
\ldf@finish{bulgarian}
⟨/code⟩
```