The GOST package

Igor Kotelnikov,* Leonid Sinev

Version ? released ?

Abstract

GOST is a bundle of BibTeX styles designed to meet State Standards (GOST) on information, librarianship and publishing issued by The Russian Federation and Interstate Committee of former USSR States.

It comprises 16 BibTeX styles to format bibliography in English, Russian and Ukrainian according to GOST 7.0.5-2008 and GOST 7.1-2003. Both 8-bit and Unicode (UTF-8) versions of each BibTeX style, in each case offering a choice of sorted and unsorted.

1 Introduction

The package was initially developed by Maksym Polyakov. It was later updated by Igor Kotelnikov to the present status and some code was borrowed from disser package developed by Stanislav Kruchinin and unpublished work by Artem Petrenkov.

Nowdays, GOST is a bundle of BibTeX styles designed to meet State Standards (GOST) on information, librarianship and publishing issued by Russian Federation and interstate committee of former USSR States.

The System of Standards includes:

GOST 7.80 -2000 Bibliographic record. Heading. General requirements and rules.

GOST 7.83 -2001 Electronic editions. Basic types and imprint.

GOST 7.1 -2003 Bibliographic record. Bibliographic description. General requirements and rules.

GOST 7.11 -2004 Bibliographic description and references. Rules for the abbreviation of words and word combinations in foreign European languages.

GOST 7.0.5-2008 Bibliographic reference. General requirements and rules of making.

Etc.

Currently, GOST contains 16 BibTeX styles to format bibliography in English, Russian and Ukrainian according to GOST 7.0.5-2008 and GOST 7.1-2003. Both 8-bit and Unicode (UTF-8) versions of each BibTeX style, in each case offering a choice of sorted and unsorted.

---

*E-mail: kia999@mail.ru

All styles in the GOST bundle are derived from single master file `gost.dtx` by applying different set of options as shown in the table below.

Style	utf8	strict	eprint	long	sort	natbib
gost2003		+	+			
gost2003s		+	+		+	
gost2008			+			
gost2008n			+			+
gost2008l			+	+		
gost2008s			+		+	
gost2008ns			+		+	+
gost2008ls			+	+	+	
ugost2003	+	+	+			
ugost2003s	+	+	+		+	
ugost2008	+		+			
ugost2008n	+		+			+
ugost2008l	+		+	+		
ugost2008s	+		+		+	
ugost2008ns	+		+		+	+
ugost2008ls	+		+	+	+	
Style	utf8	strict	eprint	long	sort	natbib

Gost2008 style is recommended for most applications. It corresponds to the currently effective Standard 7.0.5-2008. Librarians should use the style `gost2003` instead of `gost2008` to compile a library catalog to meet the Standard 7.1-2003. Use of other styles is best explained through the meaning of options used to compile those styles from the master source.

The `strict` option provides conformance to the Standard 7.1-2003. The bibstyles compiled with that option bear the name `gost2003` with possible suffixes `s`, `l`, `n` as explained below. These styles are intended primarily for the librarians who compose a library catalog.

The bibstyles compiled without `strict` option meets the Standard 7.0.5-2008 which can be thought off as a relaxed version of the Standards 7.1-2003. These bibstyles bear the name `gost2008` with possible suffixes `s`, `l`, `n`.

If the number of authors exceeds 4, modern styles cut the list of authors to at most 4 persons as prescibed by the Standards. Option `long` overrides this rule to provide backward compatibility with the package disser by Stanislab Kruchinin. Two styles, `gost2008l` and `gost2008ls`, compiled with the option `long` mimic behavior of the styles `gost705` and `gost705s` from the disser package. Major effect of the `long` option is that the list of authors always precedes book or article title no matter how long is it. Modern styles compiled without `long` place long list of authors behind the title. The names of styles compiled with the option `long` has the suffix `l`. Recall that those styles do not conform effective Standards and their use is discouraged.

The `eprint` option enables formatting electronic publications. In particular, it enables `eprint`, `eprinttype`, `eprintclass`, and `doi` fields for a bibliographic entry. The styles generated without the `eprint` option, ignore the these fields. Starting from the version 1.2 of the GOST package, all

modern styles are compiled with this option included, and the suffix `e` which designated this option in earlier versions is not appended to the name of style any more.

The `natbib` option provides compatibility with the `natbib` package. The names of styles compiled with the option `natbib` bear the suffix `n`. Currently 4 styles with that option are available for beta testing.

The `sort` option enables sorting bibliographic references by author names and references titles. The names of styles compiled with the option `sort` bear the suffix `s`.

Finally, the `utf8` option produces bibliographic styles in unicode rather that in 8-bit encoding. Names of those styles bear the prefix `u`.

Beyond bibliographic style, GOST bundle contains CS files (codepage and sorting order).

```
----------------------------------------------------------------
Encoding        |   CSF                 |  Sorting order
----------------------------------------------------------------
cp866           |   ruscii.csf          |  Cyrillic first, Latin
cp1251          |   cp1251.csf          |  Cyrillic first, Latin
koi8-u          |   koi8u.csf           |  Cyrillic first, Latin
utf8            |   utf8cyrillic.csf    |  Cyrillic first, Latin
----------------------------------------------------------------
```

In addition, BIBTEX8 distribution comes with few more CSFs.

```
----------------------------------------------------------------
Encoding        |   CSF                 | Sorting order
----------------------------------------------------------------
cp866           |   cp866rus.csf        | Latin first, Cyrillic
----------------------------------------------------------------
```

2   How to use

1. Select bibliography style by adding appropriate `\bibliographystyle` declaration to your source file `<filename>.tex`, e.g.

   ```
   \bibliographystyle{gost2008}
   \bibliography{database}
   ```

2. Add the field `langid="ukrainian"` or `langid="russian"` to the bibliographic entries in Ukrainian or Russian languages in your database; English is the default language. German, Italian and French are partially supported.

3. To compile list of references from your database use `bibtex8.exe` rather than `bibtex.exe`. Depending on the codepage of your bibliographic database, indicate one of the CS files listed above as option to `bibtex8.exe`. Run LaTeX, then run `bibTeX8` and LaTeX again:

   ```
   latex <filename>.tex
   bibtex8 -B -c <csf_file>.csf <filename>.aux
   latex <filename>.tex
   ```

4. For details on preparing bibliographic database see examples in `gost*.pdf` and `ugost*.pdf`.

5. `ugost*` styles are primarily intended for use with unicode compilers (`xelatex` and `lualatex`). They should be preferred as well when using 8bit compilers (`latex` and `pdflatex`) if source file is in utf8 encoding.

6. Neither `bibtex.exe` nor `bibtex8.exe` provides correct sorting order of unicode text. It means that using `ugost2008s` or `ugost2008ns` may produce unexpected result for documents in `utf8` encoding.

7. `Bibtex8` fails to change case of a string if it contains Cyrillic letter in unicode. Therefore `ugost2008*` styles do not change case of titles and other parts of bibliographic record while 8-bit styles do the case change where appropriate.

8. Either `bibtex.exe` or `bibtex8.exe` fail to cut Cyrillic names to initials. Therefore `ugost2008*` styles do not modify name of authors.

9. Package `natbib` is required when choosing styles with suffix `n` in their names.

3 Customization

Every GOST style defines few commands to format some parts of a reference. You can redefine these commands prior to the `\bibliography{<bibtex_style>}` command. Initial definitions are listed below.

```
\providecommand*{\url}[1]{{\small #1}}
\providecommand*{\BibUrl}[1]{\url{#1}}
\providecommand{\BibAnnote}[1]{}
\providecommand*{\BibEmph}[1]{#1}
```

By default, gost styles separate logical parts of a bibliography record by a period and cyrdash (`. "---`). It is legitimate to drop that dash by overriding the command `\BibDash` as follows

```
\providecommand*{\BibDash}{}
```

By default, `\BibDash` is equivalent to the shorthand `"---` defined by the babel package with the option `russian`. It prints a so called Cyrillic dash (`\cyrdash`), which is 20% shorter then ordinary LaTeX dash (`---`), and puts unbreakable space before `\cyrdash` so that dash never appears in the beginning of a line.

4 Where to get

1. http://ctan.org/pkg/gost.

2. http://github.com/kia999/gost.

5   Version history

Version 1.2k (2020.12.29)

1. All stuff is now generated from `gost.dtx`.

2. Limited support of the `date` field added.

3. Formatting of `doi` field updated: `http://dx.doi.org` changed to `https://doi.org`.

4. Restricted support of `date` field added.

5. Documentation and examples update (thanks to Leonid Sinev).

Version 1.2i (2017.01.12)

1. Documentation and examples update (thanks to Leonid Sinev).

2. Restored `@MastersThesis` instead of `@MasterThesis` (thanks to Leonid Sinev).

3. `media="eresource"` is introduced in addition to `media="online"` and `media="text"`; if present, the `media` field is not ignored any more in modern bst-styles compiled without the `strict` option.

4. `location` field is introduced as an alias of `address` field.

5. `@DSCISTHESIS` entry renamed to `@DOCTHESIS`.

6. `school` field in `@THESIS` and similar entries is replaced by `institution` to comply with `biblatex-gost` style.

Version 1.2h (2016.08.21)

1. Minor changes in documentation.

Version 1.2g (2016.07.25)

1. Minor changes in documentation.

Version 1.2f (2016.07.12)

1. Support for patent entry added (thanks to Stanislav Kruchinin).

2. `medium` field renamed to `media` field for compatibility with biblatex.

Version 1.2e (2016.07.07)

1. Hard coded "URL" string replaced with a language sensitive string (thanks to Roman Budnyi).

Version 1.2d (2015.02.18)

1. jan, feb, etc. macros fixed.

2. New macro `format.month`.

Version 1.2c (2015.01.10)

1. `langid` field is added. It has same meaning as `language` which is now obsolete but is still supported for backward compatibility; `langid` has priority over `language`.

2. `eid` field is added. It has priority over `pages`.

3. The ligature `"---` has been substituted with `\BibDash` for `.bst` styles compiled without `modern` options (`gost2003.bst` and `gost2003s.bst`). For modern styles this was done in earlier versions.

4. Spacing around `\BibDash` has been improved.

5. `\BibDash` now typesets short em-dash (`\cyrdash`) only for `russian` and `ukrainian` languages. In earlier versions, it produces short em-dash for all languages.

    (This feature was removed since it did not work with all engines.)

Version 1.2a (2012.08.31)

1. `\cyrdash` is now defined via `\ProvideTextCommand` rather than `\providecommand`.

Version 1.2 (2012.02.22)

1. Code refactoring. All styles are now generated from single source file.

2. Support for GOST 7.1-2003. The field `medium` is added to reflect type of material. For most entry types `medium` defaults to `text`.

3. Support for `natbib` package.

4. All modern styles are now compiled with the `eprint` option.

Version 1.1 (2012.01.21)

1. Support for GOST 7.0.5-2008 and GOST 7.1-2003 is provided.

2. `@Online` entry is added to format a reference to electronic resource on Internet.

3. `@MastersThesis` entry is added to format a reference to master's thesis.

4. `@DSciThesis` entry is added to format a reference to doctor of sciences thesis.

5. `Urldate`, `eprint`, `eprintclass`, `eprinttype` fields are added.

5.1 Older versions

2012.02.22 Support for `natbib` package.

2012.02.02 Adaptation to GOST 7.0.5, electronic publishing.

2005.08.12 First version uploaded to CTAN.

2003.06.06 First public version.

6 Implementation

We need Russian fonts to produce documentation of the code below. Therefore we switch current language to Russian by issuing the command `\selectlanguage{russian}`.

```
1  ⟨*bst⟩

2  %% This bibstyle attempts to format bibliography according to
3  ⟨strict⟩%% GOST 7.1-2003 for bibliographic records.
4  ⟨!strict⟩%% GOST 7.0.5-2008 for bibliographic reference.
5  ⟨*natbib⟩%%
6  %%-----------------------------------------------------------------
7  %% This is an author-year citation style bibliography.
8  %% It requires a special package file to function properly
9  %% such as natbib.sty by Patrick W. Daly.
10 %% The form of the \bibitem entries is
11 %%    \bibitem[Jones et al.(1990)]{key}...
12 %%    \bibitem[Jones et al.(1990)Jones, Baker, and Smith]{key}...
13 %% where the label part [in brackets] consists of the author names,
14 %% as they should appear in the citation, with the year in parentheses following.
15 %% There must be no space before the opening parenthesis!
16 %% A full list of authors may also follow the year.
17 %% In natbib.sty, it is possible to define the type of enclosures that is
18 %% really wanted (brackets or parentheses), but in either case, there must
19 %% be parentheses in the label.
20 %% The \cite command functions as follows:
21 %%    \citet{key}            => Jones et al. (1990)
22 %%    \citet*{key}           => Jones, Baker, and Smith (1990)
23 %%    \cite{key}             => (Jones et al., 1990)
24 %%    \cite*{key}            => (Jones, Baker, and Smith, 1990)
25 %%    \cite[chap. 2]{key}    => (Jones et al., 1990, chap. 2)
26 %%    \cite[e.g.][]{key}     => (e.g. Jones et al., 1990)
27 %%    \cite[e.g.][p. 32]{key} => (e.g. Jones et al., p. 32)
28 %%    \citeauthor{key}       => Jones et al.
29 %%    \citeauthor*{key}      => Jones, Baker, and Smith
30 %%    \citeyear{key}         => 1990
31 %%-----------------------------------------------------------------
32 ⟨/natbib⟩
33
```

6.1 Fields

Enlist all entry fields allowed in a bibliographic database. Most fields are common for many standard bst styles. Nonlisted fields are just ignored by BiBTeX.

```
34 ENTRY
35   { address
36     annote
37     author
38     booktitle
39     bookauthor
40     chapter
41     edition
42     editor
43     compiler
44     howpublished
45     institution
46     journal
47     key
48     %major          % new in v.1.2i, alias for speciality, not implemented
49     majorcode       % new in v.1.2i, alias for specialitycode
50     month
51     note
52     number
53     organization
54     pages
55     eid             % new in v1.2c
56     publisher
57     school          % alias for institution
58     series
59     %speciality     % new in v.1.2i, eqv. to major in biblatex-gost, not implemented yet...
60     specialitycode  % new in v.1.2i, alias of number, eqv. to majorcode in biblatex-gost
61     title
62     %medium         % new in v1.2; renamed to media.
63     media           % new in v1.2f
64     type
65     volume
66     year
67     language
68     langid          % new in v1.2c
69     booklanguage
```

Entries borrowed from biblatex.

```
70     date            % new in v1.2i
71     pagetotal
72     url
73     urldate
74     isbn
75     doi
76     eprint
77     eprinttype      % = archivePrefix
78     eprintclass     % = primaryClass
```

```
 79     % new in v1.2f:
 80     % appear in biblatex:
 81     %addendum        % not implemented yet...
 82     holder           % see patent
 83     location         % new in v.1.2i, alias of address
 84     %subtitle        % not implemented yet...
 85     titleaddon       % new in v.1.2i, see @thesis
 86     %version         % not implemented yet...
 87     % Appear in biblatex-gost for @patent entry:
 88     authorcountry    % country of the patent authors
 89     credits          % statement of responsibility, other than provided in Biblatex
 90     ipc              % Code of the International Patent Classification
 91     %media           % General material designation NOTE: medium in the above
 92     requestnumber    % Registration number of the application to the patent document
 93     publicationdate % Date of publication
 94     publication      %  and information on the official gazette, which published patent
 95     prioritydate     % Information about the convention priority: the date of filing of the application,
 96     prioritynumber  %   number and
 97     prioritycountry %   country name of convention priority.
 98     requestdate      % ??
 99   }
100   {}
101 ⟨!natbib⟩   { label }
102 ⟨natbib⟩    { label extra.label sort.label short.list }
103
```

6.2   Output functions

Declare internal variables and constants used for formatting bibliographic records. Other variables
are defined below when needed.

```
104 INTEGERS {
105     output.state
106     before.all
107     mid.sentence
108     after.sentence
109     after.block
110     after.dblslash
111     after.slash
112     after.colon
113     after.semicolon
114 }
115
116 STRINGS { curlanguage }
117
118 STRINGS { s t }
119
120 STRINGS { y m d }   % new in v.1.2j
121
```

init.state.consts   Set constants that designate various output states which are kept by 'output.state' integer. The

9

latter is checked and updated by a set of functions such as 'output' to be defined below.

```
122 FUNCTION {init.state.consts}
123 { #0 'before.all :=
124   #1 'mid.sentence :=
125   #2 'after.sentence :=
126   #3 'after.block :=
127   #4 'after.dblslash :=
128   #5 'after.slash :=
129   #6 'after.colon :=
130   #7 'after.semicolon :=
131 }
132
```

set.language   Sets current language `curlanguage`. Called by `bibitem.output` before any other function.

```
133 FUNCTION {set.language}
134 { langid empty$
135     { language empty$
136         { "english" 'curlanguage := }
137         { language  'curlanguage := }
138       if$
139     }
140     { langid  'curlanguage := }
141   if$
142 }
143
```

reset.language   Reset current language to `booklanguage` if provided. Called by `output.nonnull` after double slash so that the rest of the record is formatted with `booklanguage`.

```
144 FUNCTION {reset.language}
145 { booklanguage empty$
146     { "" }
147     { booklanguage  'curlanguage :=
148       "\selectlanguageifdefined{"
149       curlanguage *
150       "}" *
151     }
152   if$
153 }
154
```

Declare various functions to output various parts of a bibliographic record.

output.nonnull   Writes the last literal in the stack to output buffer assuming that it is not empty and adds an appropriate punctuation symbol.

```
155 FUNCTION {output.nonnull}
156 {
157   swap$
158   output.state mid.sentence =
159     { ", " * write$ }
160     { output.state after.block =
161         { add.period$ write$
162             " \BibDash " write$
```

```
163         newline$
164         "\newblock " write$
165       }
166     { output.state before.all =
167         'write$
168       { output.state after.dblslash =
169           { "~//" * reset.language * " " * write$ }
170         { output.state after.slash =
171             { "~/ " * write$ }
172           { output.state after.colon =
173               { "~: " * write$ }
174             { output.state after.semicolon =
175                 { "~; " * write$ }
176             { add.period$ " " * write$ }
177               if$
178               }
179             if$
180           }
181         if$
182       }
183     if$
184   }
185   if$
186 }
187 if$
188 mid.sentence 'output.state :=
189   }
190 if$
191 }
192
```

output   Calls `output.nonnull` if the last literal string in the stack is not empty; otherwise it discards the literal.

```
193 FUNCTION {output}
194 { duplicate$ empty$
195     'pop$
196     'output.nonnull
197   if$
198 }
199
```

output.check   Does the same but also warns if the indicated field is empty. Needs two literals in the stack: the field and the name of the field, e.g., `author "author" output.check`.

```
200 FUNCTION {output.check}
201 { 't :=
202   duplicate$ empty$
203     { pop$
204       "empty " t * " in " * cite$ * warning$
205     }
206     'output.nonnull
207   if$
```

```
208 }
209
```

fin.entry   `fin.entry` finalizes current entry. It writes dot, if no dot is found in stack, and starts new line.

```
210 FUNCTION {fin.entry}
211 { add.period$
212   write$
213   newline$
214 }
215
```

      Declare a family of functions to put punctuation marks depending on the current status of the output stack.

new.block   This just checks output state and revert it to another state if required. Checking output state prevents occasional doubling of punctuation marks.

```
216 FUNCTION {new.block}
217 { output.state before.all =
218     'skip$
219     { after.block 'output.state := }
220   if$
221 }
222
```

new.dblslash

```
223 FUNCTION {new.dblslash}
224 { output.state before.all =
225     'skip$
226     { after.dblslash 'output.state := }
227   if$
228 }
229
```

new.slash

```
230 FUNCTION {new.slash}
231 { output.state before.all =
232     'skip$
233     { after.slash 'output.state := }
234   if$
235 }
236
```

new.colon

```
237 FUNCTION {new.colon}
238 { output.state before.all =
239     'skip$
240     { after.colon 'output.state := }
241   if$
242 }
243
```

new.semicolon

```
244 FUNCTION {new.semicolon}
```

```
245 { output.state before.all =
246     'skip$
247     { after.semicolon 'output.state := }
248   if$
249 }
250
```

new.sentence

```
251 FUNCTION {new.sentence}
252 { output.state after.block =
253     'skip$
254     { output.state before.all =
255         'skip$
256         { after.sentence 'output.state := }
257       if$
258     }
259   if$
260 }
261
```

add.blank

```
262 FUNCTION {add.blank}
263 { " " * before.all 'output.state :=
264 }
265
```

6.3   Logical functions and various checks

Declare few logical functions.

not

```
266 FUNCTION {not}
267 {   { #0 }
268     { #1 }
269   if$
270 }
271
```

and

```
272 FUNCTION {and}
273 {   'skip$
274     { pop$ #0 }
275   if$
276 }
277
```

or

```
278 FUNCTION {or}
279 {   { pop$ #1 }
280     'skip$
281   if$
282 }
283
```

non.stop   ⟨NB: What's the hell? Never used.⟩

```
284 %FUNCTION {non.stop}
285 %{ duplicate$
286 %    "}" * add.period$
287 %    #-1 #1 substring$ "." =
288 %}
289 %
```

new.block.checka   Adds `new.block` if the last literal in stack is not empty.

```
290 FUNCTION {new.block.checka}
291 { empty$
292     'skip$
293     'new.block
294   if$
295 }
296
```

new.block.checkb   Adds `new.block` if either of the two last literals in the stack is not empty.

```
297 FUNCTION {new.block.checkb}
298 { empty$
299   swap$ empty$
300   and
301     'skip$
302     'new.block
303   if$
304 }
305
```

new.sentence.checka   Adds `new.sentence` if the last literal in stack is not empty.

```
306 FUNCTION {new.sentence.checka}
307 { empty$
308     'skip$
309     'new.sentence
310   if$
311 }
312
```

new.sentence.checkb   Adds `new.sentence` if either of the two last literals in the stack is not empty.

```
313 FUNCTION {new.sentence.checkb}
314 { empty$
315   swap$ empty$
316   and
317     'skip$
318     'new.sentence
319   if$
320 }
321
```

new.dblslash.checka   For `online` entry.

```
322 FUNCTION {new.dblslash.checka}
323 { empty$
324     'skip$
```

14

```
325     'new.dblslash
326   if$
327 }
328
```

field.or.null   Replaces an empty field with null string "".

```
329 FUNCTION {field.or.null}
330 { duplicate$ empty$
331     { pop$ "" }
332     'skip$
333   if$
334 }
335
```

either.or.check   ⟨NB: Move upwards⟩

```
336 FUNCTION {either.or.check}
337 { empty$
338     'pop$
339     { "can't use both " swap$ * " fields in " * cite$ * warning$ }
340   if$
341 }
342
```

6.4   String Functions

spaces.around   Inserts a space before and after last literal in the stack.

```
343 FUNCTION {spaces.around}
344 { " " swap$ * " " * }
345
```

emphasize   Emphasizes the last literal in the stack if it is not empty. v1.2k: `emphasize` now returns empty field if last literal in the stack was empty.

```
346 FUNCTION {emphasize}
347 { duplicate$ empty$
348     %{ pop$ "" }
349     'skip$        % v.1.2k
350     { "\BibEmph{" swap$ * "}" * }
351   if$
352 }
353
```

bracify   New in v.1.2. An idea borrowed from apsrev4-1.bst. Encloses last literal in the stack by braces even if it is empty. Note that braces are normally not printed by LaTeX.

```
354 FUNCTION {bracify}
355 { duplicate$ empty$
356     { pop$ "{}" }
357     { "{" swap$ * "}" * }
358   if$
359 }
360
```

bracketise This and the next functions are used to enclose last word by square and round brackets. In contrast to `bracify` function they push null string if the last literal is empty.

```
361 FUNCTION {bracketise}
362 {
363   duplicate$ empty$
364     { pop$ "" }
365     { "[" swap$ * "]" * }
366   if$
367 }
368
```

paranthesify

```
369 FUNCTION {paranthesify}
370 {
371   duplicate$ empty$
372     { pop$ "" }
373     { "(" swap$ * ")" * }
374   if$
375 }
376
```

chop.word

The function `chop.word` in the context '*sstr len str* `chop.word`' tries to remove given substring *sstr* of the length *len* from the beginning of the string *str*. It trims *str* only if first *len* symbols in *str* coinsides with *sstr*. See examples in Section 6.8.

```
377 ⟨∗sort | natbib⟩
378 INTEGERS { len }
379
380 FUNCTION {chop.word}
381 { 's :=
382   'len :=
383   s #1 len substring$ =
384     { s len #1 + global.max$ substring$ }
385     's
386   if$
387 }
388 ⟨/sort | natbib⟩
389
```

is.num Currently not used.
⟨NB: Можно использовать для проверки года, извлечённого из `date`.⟩

```
390 ⟨∗debug⟩
391 FUNCTION {is.num}
392 { chr.to.int$
393   duplicate$ "0" chr.to.int$ < not
394   swap$ "9" chr.to.int$ > not and
395 }
396
397 ⟨/debug⟩
```

extract.num Currently not used.

```
398 ⟨∗debug⟩
399 FUNCTION {extract.num}
400 { duplicate$ 't :=
401   "" 's :=
402   { t empty$ not }
403   { t #1 #1 substring$
404     t #2 global.max$ substring$ 't :=
405     duplicate$ is.num
406       { s swap$ * 's := }
407       { pop$ "" 't := }
408     if$
409   }
410   while$
411   s empty$
412     'skip$
413     { pop$ s }
414   if$
415 }
416
417 ⟨/debug⟩
```

tie.connect Inserts unbreakable space between last two literals in the stack.

```
418 FUNCTION {tie.connect}
419 {"~" swap$ * *
420 }
421
```

.or.space.connect Inserts space or unbreakable space between last two literals in the stack depending on the length of last literal.

```
422 FUNCTION {tie.or.space.connect}
423 { duplicate$ text.length$ #3 <
424     { "~" }
425     { " " }
426   if$
427   swap$ * *
428 }
429
```

n.dashify

```
430 FUNCTION {n.dashify}
431 { 't :=
432   ""
433     { t empty$ not }
434     { t #1 #1 substring$ "-" =
435         { t #1 #2 substring$ "--" = not
436             { "--" *
437               t #2 global.max$ substring$ 't :=
438             }
439             {   { t #1 #1 substring$ "-" = }
440                 { "-" *
441                   t #2 global.max$ substring$ 't :=
```

17

```
442                        }
443                    while$
444                }
445            if$
446        }
447        { t #1 #1 substring$ *
448          t #2 global.max$ substring$ 't :=
449        }
450      if$
451    }
452  while$
453 }
454
```

multi.page.check   Returns 1 if the last literal (usually `page`) contains either '`-`', '`,`' or '`+`'; otherwise returns 0. Used in connection with `n.dashify`.

⟨NB: Заметим несогласованность функций `multi.page.check` и `n.dashify`. Последняя функция проверяет дефис, а первая ещё и минус и запятую.⟩

```
455 INTEGERS { multiresult }
456
457 FUNCTION {multi.page.check}
458 { 't :=
459   #0 'multiresult :=
460     { multiresult not
461       t empty$ not
462       and
463     }
464     { t #1 #1 substring$
465       duplicate$ "-" =
466       swap$ duplicate$ "," =
467       swap$ "+" =
468       or or
469         { #1 'multiresult := }
470         { t #2 global.max$ substring$ 't := }
471       if$
472     }
473   while$
474   multiresult
475 }
476
```

6.5 Language-sensitive abbreviations

Declare language-sensitive abbreviations. We provide two versions of any abbreviation for unicode and non-Unicode styles. The language-sensitive functions push to the stack a string that depends of the the current value of the string `curlanguage`. It is set for every entry by `output.bibitem` function. Abbreviations for `russian`, `ukrainian` and `english` values of the string `curlanguage` are always provided, and sometimes for `french` and `german`.

bbl.edby

```
477 FUNCTION {bbl.edby}
478 { curlanguage "english" =
479    {"ed.\ by"}
480    { curlanguage "ukrainian" =
481 ⟨!utf8⟩        {"{\cyr\cyrp\cyrii\cyrd\ \cyrr\cyre\cyrd.}"}
482 ⟨utf8⟩         {"під ред."}
483       { curlanguage "russian" =
484 ⟨!utf8⟩           {"{\cyr\cyrp\cyro\cyrd\ \cyrr\cyre\cyrd.}"}
485 ⟨utf8⟩            {"под ред."}
486          { curlanguage "german" =
487             { "ed." }
488             {"language is not defined: " curlanguage " in bbl.edby" * * warning$ "Ed.\ by"}
489          if$}
490       if$}
491    if$}
492 if$}
493
```

```
494 FUNCTION {bbl.compiler}
495 { curlanguage "english" =
496    { "Compiler"}
497    { curlanguage "german" =
498       { "Hrsg." }
499       { curlanguage "ukrainian" =
500 ⟨!utf8⟩           {"{\cyr\CYRU\cyrk\cyrl.}"}
501 ⟨utf8⟩            {"{Укл.}"}
502          { curlanguage "russian" =
503 ⟨!utf8⟩              {"{\cyr\CYRS\cyro\cyrs\cyrt.}"}
504 ⟨utf8⟩               {"{Сост.}"}
505             {"language is not defined: " curlanguage  " in bbl.compiler" * * warning$ "Compiler"}
506          if$}
507       if$}
508    if$}
509 if$}
510
```

```
511 FUNCTION {bbl.edition}
512 { curlanguage "english" =
513    {"ed."}
514    { curlanguage "ukrainian" =
515 ⟨!utf8⟩        {"{\cyr\cyrv\cyri\cyrd.}"}
516 ⟨utf8⟩         {"{вид.}"}
517       { curlanguage "russian" =
518 ⟨!utf8⟩           {"{\cyr\cyri\cyrz\cyrd.}"}
519 ⟨utf8⟩            {"{изд.}"}
520          { curlanguage "german" =
521             {" {aus.}" } %%%% { "Aufl." } ??
522             { curlanguage "italian" =
523                {"edizione"}
```

```
524            { curlanguage "french" =
525                {"\'{e}dition"}
526                {"language is not defined: " curlanguage  " in bbl.edition" * * warning$ "ed."}
527              if$}
528            if$}
529          if$}
530        if$}
531      if$}
532 if$}
533
```

bbl.vvolume

```
534 FUNCTION {bbl.vvolume}
535 { curlanguage "english" = curlanguage "french" = or curlanguage "italian" = or
536    {"Volume"}
537    { curlanguage "ukrainian" = curlanguage "russian" = or
538 ⟨!utf8⟩      { "\CYRT\cyro\cyrm" }
539 ⟨utf8⟩       { "Том" }
540      { curlanguage "german" =
541         {"{Band}"} %%%% { "Volumen" }
542         {"language is not defined: " curlanguage  " in bbl.vvolume" * * warning$ "Volume"}
543        if$}
544     if$}
545 if$}
546
```

bbl.vvol

```
547 FUNCTION {bbl.vvol}
548 { curlanguage "english" = curlanguage "french" = or curlanguage "italian" = or
549    {"Vol."}
550    { curlanguage "ukrainian" = curlanguage "russian" = or
551 ⟨!utf8⟩      {"\CYRT."}
552 ⟨utf8⟩       {"Т."}
553      { curlanguage "german" =
554         {"{Bd.}"} %%%% { "Vol." }
555         {"language is not defined: " curlanguage  " in bbl.vvol" * * warning$ "Vol."}
556        if$}
557     if$}
558 if$}
559
```

bbl.iissue

```
560 FUNCTION {bbl.iissue}
561 { curlanguage "english" =
562    {"Issue"}
563    { curlanguage "ukrainian" =
564 ⟨!utf8⟩      {"\CYRV\cyri\cyrp\cyru\cyrs\cyrk"}
565 ⟨utf8⟩      {"Випуск"}
566      { curlanguage "russian" =
567 ⟨!utf8⟩        {"\CYRV\cyrery\cyrp\cyru\cyrs\cyrk"}
568 ⟨utf8⟩        {"Выпуск"}
569        { curlanguage "german" =
```

```
570             {"{Heft}"} %%%% { "Ausgabe" }
571             {"language is not defined: " curlanguage " in bbl.iissue" * * warning$ "Issue"}
572         if$}
573       if$}
574     if$}
575 if$}
576
```

bbl.iiss

```
577 FUNCTION {bbl.iiss}
578 { curlanguage "english" =
579     {"Iss."}
580     { curlanguage "ukrainian" =
581 ⟨!utf8⟩       {"\CYRV\cyri\cyrp."}
582 ⟨utf8⟩        {"Вип."}
583       { curlanguage "russian" =
584 ⟨!utf8⟩           {"\CYRV\cyrery\cyrp."}
585 ⟨utf8⟩            {"Вып."}
586         { curlanguage "german" =
587             {"{H.}"}
588             {"language is not defined: " curlanguage " in bbl.iiss" * * warning$ "Iss."}
589         if$}
590       if$}
591     if$}
592 if$}
593
```

bbl.of

```
594 FUNCTION {bbl.of}
595 { curlanguage "english" =
596     {"of"}
597     { curlanguage "german" =
598         { "von" }
599         { curlanguage "ukrainian" =
600 ⟨!utf8⟩           { "{\cyr\cyrii\cyrz}" }
601 ⟨utf8⟩            { "{iз}" }
602           { curlanguage "russian" =
603 ⟨!utf8⟩               { "{\cyr\cyri\cyrz}" }
604 ⟨utf8⟩                { "{из}" }
605             {"language is not defined: " curlanguage " in bbl.of" * * warning$ "of"}
606         if$}
607       if$}
608     if$}
609 if$}
610
```

bbl.etal

```
611 FUNCTION {bbl.etal}
612 { curlanguage "english" =
613     {"et~al."}
614     { curlanguage "german" =
615         { "u.~a." }
```

```
616        { curlanguage "ukrainian" =
617 ⟨!utf8⟩           {"{\cyr\cyrt\cyra~\cyrii\cyrn.}"}
618 ⟨utf8⟩            {"{та~iн.}"}
619          { curlanguage "russian" =
620 ⟨!utf8⟩             {"{\cyr\cyri~\cyrd\cyrr.}"}
621 ⟨utf8⟩              {"и~др."}
622             {"language is not defined: " curlanguage  " in bbl.etal" * * warning$ "et~al."}
623          if$}
624        if$}
625     if$}
626 if$}
627
```

bbl.and

```
628 FUNCTION {bbl.and}
629 { curlanguage "english" =
630   {"and"}
631     { curlanguage "german" =
632        { "und" }
633        { curlanguage "ukrainian" =
634 ⟨!utf8⟩          {"{\cyrii}"}
635 ⟨utf8⟩           {"i"}
636          { curlanguage "russian" =
637 ⟨!utf8⟩             {"{\cyri}"}
638 ⟨utf8⟩              {"и"}
639            { curlanguage "french" =
640                {"et"}
641                {"language is not defined: " curlanguage  " in bbl.and" * * warning$ "and"}
642            if$}
643          if$}
644        if$}
645     if$}
646 if$}
647
```

bbl.nnumber

```
648 FUNCTION {bbl.nnumber}
649 { curlanguage "english" =
650    {"Number"}
651     { curlanguage "ukrainian" = curlanguage "russian" = or
652 ⟨!utf8⟩       { "\CYRN\cyro\cyrm\cyre\cyrr" }
653 ⟨utf8⟩       { "{Номер}" }
654       { curlanguage "german" =
655          {"{Heft}"} %%% { "Anzahl" }
656          {"language is not defined: " curlanguage  " in bbl.nnumber" * * warning$ "Number"}
657       if$}
658    if$}
659 if$}
660
```

bbl.number

```
661 FUNCTION {bbl.number}
```

```
662 { curlanguage "english" =
663    {"number"}
664    { curlanguage "ukrainian" = curlanguage "russian" = or
665 ⟨!utf8⟩      {"{\cyr\cyrn\cyro\cyrm\cyre\cyrr}"}
666 ⟨utf8⟩       {"{номер}"}
667      { curlanguage "german" =
668        {"{Heft}"} %%% { "anzahl" }???
669        {"language is not defined: " curlanguage  " in bbl.number" * * warning$ "number"}
670      if$}
671    if$}
672 if$}
673
```

bbl.nr

```
674 FUNCTION {bbl.nr}
675 { curlanguage "english" =
676    {"no."}
677    { curlanguage "italian" =
678        { "no." }
679        { curlanguage "ukrainian" = curlanguage "russian" = or
680 ⟨!utf8⟩        { "{\cyr\textnumero}" }
681 ⟨utf8⟩         { "{№}" }
682          { curlanguage "german" =
683            {"{nu.}"} %%% { "an." }
684            { curlanguage "french" =
685                { "no." }
686                {"language is not defined: " curlanguage " in bbl.nr" * * warning$ "no."}
687            if$}
688          if$}
689        if$}
690    if$}
691 if$}
692
```

bbl.nnr

```
693 FUNCTION {bbl.nnr}
694 { curlanguage "english" =
695    {"No."}
696    { curlanguage "ukrainian" = curlanguage "russian" = or
697 ⟨!utf8⟩      { "{\cyr\textnumero}" }
698 ⟨utf8⟩       { "{№}" }
699      { curlanguage "german" =
700        {"{H.}"} %%% { "an." }
701        {"language is not defined: " curlanguage  " in bbl.nnr" * * warning$ "No."}
702      if$}
703    if$}
704 if$}
705
```

bbl.in

```
706 FUNCTION {bbl.in}
707 { curlanguage "english" = curlanguage "german" = or
```

```
708      {"in"}
709      { curlanguage "ukrainian" = curlanguage "russian" = or
710 ⟨!utf8⟩      { "{\cyr\cyrv}" }
711 ⟨utf8⟩       { "{в}" }
712          {"language is not defined: " curlanguage  " in bbl.in" * * warning$ "in"}
713      if$}
714 if$}
715
```

bbl.iin   Currently not used.

```
716 FUNCTION {bbl.iin}
717 { curlanguage "english" = curlanguage "german" = or
718      {"In"}
719      { curlanguage "ukrainian" = curlanguage "russian" = or
720 ⟨!utf8⟩      { "\CYRV" }
721 ⟨utf8⟩       { "{В}" }
722          {"language is not defined: " curlanguage  " in bbl.iin" * * warning$ "In"}
723      if$}
724 if$}
725
```

bbl.pages

```
726 FUNCTION {bbl.pages}
727 { curlanguage "english" = curlanguage "french" = or curlanguage "italian" = or
728      {"p."} %%% {"pp."}
729      { curlanguage "ukrainian" = curlanguage "russian" = or
730 ⟨!utf8⟩      {"{\cyr\cyrs.}"}
731 ⟨utf8⟩       {"{с.}"}
732          { curlanguage "german" =
733            {"S."} %%%% { "s." }
734            {"language is not defined: " curlanguage  " in bbl.pages" * * warning$ "p."}
735          if$}
736      if$}
737 if$}
738
```

bbl.page

```
739 FUNCTION {bbl.page}
740 { curlanguage "english" = curlanguage "french" = or curlanguage "italian" = or
741      {"p."}
742      { curlanguage "ukrainian" = curlanguage "russian" = or
743 ⟨!utf8⟩      {"{\cyr\cyrs.}"}
744 ⟨utf8⟩       {"{с.}"}
745          { curlanguage "german" =
746            {"S."} %%%% { "s." }
747            {"language is not defined: " curlanguage  " in bbl.page" * * warning$ "p."}
748          if$}
749      if$}
750 if$}
751
```

bbl.ppages

```
752 FUNCTION {bbl.ppages}
753 { curlanguage "english" = curlanguage "french" = or curlanguage "italian" = or
754    {"P."} %%%% { "Pp." }
755    { curlanguage "ukrainian" = curlanguage "russian" = or
756 ⟨!utf8⟩      {"{\cyr\CYRS.}"}
757 ⟨utf8⟩       {"{C.}"}
758       { curlanguage "german" =
759          {"S."}
760          {"language is not defined: " curlanguage " in bbl.ppages" * * warning$ "P."}
761       if$}
762    if$}
763 if$}
764
```

```
765 FUNCTION {bbl.ppage}
766 { curlanguage "english" = curlanguage "french" = or curlanguage "italian" = or
767    {"P."}
768    { curlanguage "ukrainian" = curlanguage "russian" = or
769 ⟨!utf8⟩      {"{\cyr\CYRS.}"}
770 ⟨utf8⟩       {"{C.}"}
771       { curlanguage "german" =
772          {"S."}
773          {"language is not defined: " curlanguage  " in bbl.ppage" * * warning$ "P."}
774       if$}
775    if$}
776 if$}
777
```

Added in version 2016.07.07.

```
778 FUNCTION {bbl.url}
779 { curlanguage "english" =
780    {"Access mode"}
781    { curlanguage "ukrainian" =
782 ⟨!utf8⟩      { "{\CYRR\cyre\cyrzh\cyri\cyrm\ \cyrd\cyro\cyrs\cyrt\cyru\cyrp\cyru}" }
783 ⟨utf8⟩       { "{Режим доступу}" }
784       { curlanguage "russian" =
785 ⟨!utf8⟩          { "{\CYRR\cyre\cyrzh\cyri\cyrm\ \cyrd\cyro\cyrs\cyrt\cyru\cyrp\cyra}" }
786 ⟨utf8⟩          { "{Режим доступа}" }
787          { curlanguage "german" =
788             { "{online; abgerufen}" }
789             { curlanguage "french" =
790                { "Mode d'acc\'{e}s" }
791                { "language is not defined: " curlanguage " in bbl.url" * * warning$ "online; accessed" }
792             if$}
793          if$}
794       if$}
795    if$}
796 if$}
```

Added in version 2012.01.15.

```
797 FUNCTION {bbl.urldate}
```

```
798 { curlanguage "english" =
799    {"online; accessed"}
800    { curlanguage "ukrainian" =
801 ⟨!utf8⟩       { "{\cyrd\cyra\cyrt\cyra\ \cyrz\cyrv\cyre\cyrr\cyrn\cyre\cyrn\cyrn\cyrya}" }
802 ⟨utf8⟩       { "{дата звернення}" }
803       { curlanguage "russian" =
804 ⟨!utf8⟩          { "{\cyrd\cyra\cyrt\cyra\ \cyro\cyrb\cyrr\cyra\cyrshch\cyre\cyrn\cyri\cyrya}" }
805 ⟨utf8⟩          { "{дата обращения}" }
806          { curlanguage "german" =
807            { "{online; abgerufen}" }
808            { curlanguage "french" =
809               { "{en ligne; acc\'{e}d\'{e}}" }
810               { "language is not defined: " curlanguage " in bbl.urldate" * * warning$ "online; accessed" ]
811            if$}
812          if$}
813       if$}
814    if$}
815 if$}
816
```

```
817 FUNCTION {bbl.techreport}
818 { curlanguage "english" =
819    { "Rep." }
820    { curlanguage "german" =
821       { "Bericht" }
822       { curlanguage "russian" =
823 ⟨!utf8⟩          { "{\cyr\CYRO\cyrt\cyrch\cyre\cyrt}" }
824 ⟨utf8⟩          { "{Отчет}" }
825          { "language is not defined: " curlanguage " in bbl.techrep" * * warning$ "Rep." }
826       if$}
827    if$}
828 if$}
829
```

```
830 FUNCTION {bbl.mathesis}
831 { curlanguage "english" =
832    { "Master's thesis" }
833    { curlanguage "german" =
834       { "diss.~mag." }
835       { curlanguage "russian" =
836 ⟨!utf8⟩          { "{\cyr\cyrk\cyrv\cyra\cyrl\cyri\cyrf\cyri\cyrk\cyra\cyrc\cyri"
837 ⟨!utf8⟩             "\cyro\cyrn\cyrn\cyra\cyrya\ \cyrr\cyra\cyrb\cyro\cyrt\cyra\ " *
838 ⟨!utf8⟩             "\cyrm\cyra\cyrg\cyri\cyrs\cyrt\cyrr\cyra}" * }
839 ⟨utf8⟩          { "{квалификационная работа магистра}" }
840          { "language is not defined: " curlanguage " in bbl.mthesis" * * warning$ "Master's thesis" }
841       if$}
842    if$}
843 if$}
844
```

26

bbl.phdthesis

```
845 FUNCTION {bbl.phdthesis}
846 { curlanguage "english" =
847    { "Ph.\,D. thesis" }
848    { curlanguage "german" =
849        { "diss.~Ph.\,D." }
850        { curlanguage "russian" =
851 ⟨!utf8⟩          { "{\cyr\cyrd\cyri\cyrs.\ \ldots\ \cyrk\cyra\cyrn\cyrd. "
852 ⟨!utf8⟩            "\cyrn\cyra\cyru\cyrk}" * }
853 ⟨utf8⟩          { "{дис.\ \ldots\ канд.\ наук}" }
854          { curlanguage "french" =
855            { "th\`{e}se de doctorat" }
856            { "language is not defined: " curlanguage " in bbl.phdthesis" * * warning$ "Ph.\,D. thesis" }
857          if$}
858        if$}
859    if$}
860 if$}
861
```

bbl.docthesis

```
862 FUNCTION {bbl.docthesis}
863 { curlanguage "english" =
864    { "dr.\,sci. dissertation" }
865    { curlanguage "german" =
866        { "diss.~dr." }
867        { curlanguage "russian" =
868 ⟨!utf8⟩          { "{\cyr\cyrd\cyri\cyrs.\ \ldots\ \cyrd-\cyrr\cyra\ "
869 ⟨!utf8⟩            "\cyrn\cyra\cyru\cyrk}" * }
870 ⟨utf8⟩          { "{дис.\ \ldots\ д-ра наук}" }
871          { "language is not defined: " curlanguage " in bbl.docthesis" * * warning$ "Dr.\,Sci. dissertation"
872        if$}
873    if$}
874 if$}
875
```

bbl.nnoaddress

```
876 FUNCTION {bbl.nnoaddress}
877 { curlanguage "english" =
878    { "S.\ l." }
879    { curlanguage "russian" =
880 ⟨!utf8⟩      { "{\cyr\CYRB.\ \cyrm.}" }
881 ⟨utf8⟩      { "{Б.\ м.}" }
882      { "language is not defined: " curlanguage " in bbl.nnoaddress" * * warning$ "S.\ l." }
883    if$}
884 if$}
885
```

bbl.nopublisher

```
886 FUNCTION {bbl.nopublisher}
887 { curlanguage "english" =
888    { "s.\ n." }
```

```
889     { curlanguage "russian" =
890 ⟨!utf8⟩        { "{\cyr\cyrb.\ \cyri.}" }
891 ⟨utf8⟩        { "{б.\ и.}" }
892        { "language is not defined: " curlanguage " in bbl.nnopublisher" * * warning$ "s.\ n." }
893     if$}
894 if$}
895
```

bbl.nnopublisher

```
896 FUNCTION {bbl.nnopublisher}
897 { curlanguage "english" =
898     { "S.\ n." }
899     { curlanguage "russian" =
900 ⟨!utf8⟩        { "{\cyr\CYRB.\ \cyri.}" }
901 ⟨utf8⟩        { "{Б.\ и.}" }
902        { "language is not defined: " curlanguage " in bbl.nnopublisher" * * warning$ "S.\ n." }
903     if$}
904 if$}
905
```

bbl.media.text

```
906 FUNCTION {bbl.media.text}
907 { curlanguage "english" =
908     { "Text" }
909     { curlanguage "russian" = curlanguage "ukrainian" = or
910 ⟨!utf8⟩        { "{\cyr\CYRT\cyre\cyrk\cyrs\cyrt}" }
911 ⟨utf8⟩        { "{Текст}" }
912        { "language is not defined: " curlanguage " in bbl.media" * * warning$ "Text" }
913     if$}
914 if$}
915
```

bbl.media.eresource

```
916 FUNCTION {bbl.media.eresource}
917 { curlanguage "english" =
918     { "Electronic resource" }
919     { curlanguage "russian" =
920 ⟨!utf8⟩        { "{\cyr\CYREREV\cyrl\cyre\cyrk\cyrt\cyrr\cyro\cyrn\cyrn\cyrery\cyrishrt\ "
921 ⟨!utf8⟩           "\cyrr\cyre\cyrs\cyru\cyrr\cyrs}" * }
922 ⟨utf8⟩        { "{Электронный ресурс}" }
923        { curlanguage "ukrainian" =
924 ⟨!utf8⟩           { "{\cyr\CYRE\cyrl\cyre\cyrk\cyrt\cyrr\cyro\cyrn\cyrn\cyri\cyrishrt\ "
925 ⟨!utf8⟩              "\cyrr\cyre\cyrs\cyru\cyrr\cyrs}" * }
926 ⟨utf8⟩           { "{Електронний ресурс}" }
927           { "language is not defined: " curlanguage " in bbl.media" * * warning$ "Electronic resource" }
928        if$}
929     if$}
930 if$}
931
```

bbl.media.online

```
932 FUNCTION {bbl.media.online}
```

28

```
933 { curlanguage "english" =
934     { "Electronic resource online" }
935     { curlanguage "russian" =
936 ⟨!utf8⟩         { "{\cyr\CYREREV\cyrl\cyre\cyrk\cyrt\cyrr\cyro\cyrn\cyrn\cyrery\cyrishrt\ "
937 ⟨!utf8⟩           "\cyrr\cyre\cyrs\cyru\cyrr\cyrs\ \cyro\cyrn\cyrl\cyra\cyrishrt\cyrn}" * }
938 ⟨utf8⟩          { "{Электронный ресурс онлайн}" }
939       { curlanguage "ukrainian" =
940 ⟨!utf8⟩           { "{\cyr\CYRE\cyrl\cyre\cyrk\cyrt\cyrr\cyro\cyrn\cyrn\cyri\cyrishrt\ "
941 ⟨!utf8⟩             "\cyrr\cyre\cyrs\cyru\cyrr\cyrs\ \cyro\cyrn\cyrl\cyra\cyrishrt\cyr}" * }
942 ⟨utf8⟩           { "{Електронний ресурс онлайн}" }
943         { "language is not defined: " curlanguage " in bbl.media" * * warning$ "Electronic resource" }
944       if$}
945     if$}
946 if$}
947
```

bbl.chief

```
948 FUNCTION {bbl.chief}
949 { curlanguage "english" =
950     { "chief" }
951     { curlanguage "russian" =
952 ⟨!utf8⟩         { "\cyrr\cyru\cyrk." }
953 ⟨utf8⟩          { "{рук.}" }
954       { curlanguage "ukrainian" =
955 ⟨!utf8⟩           { "\cyrr\cyru\cyrk." }
956 ⟨utf8⟩            { "{рук.}" }
957         { "language is not defined: " curlanguage " in bbl.chief" * * warning$ "chief" }
958       if$}
959     if$}
960 if$}
961
```

bbl.executor

```
962 FUNCTION {bbl.executor}
963 { curlanguage "english" =
964     { "executor" }
965     { curlanguage "russian" =
966 ⟨!utf8⟩         { "{\cyr\cyri\cyrs\cyrp\cyro\cyrl\cyrn.}" }
967 ⟨utf8⟩          { "{исполн.}" }
968       { curlanguage "ukrainian" =
969 ⟨!utf8⟩            { "{\cyr\cyrv\cyri\cyrk\cyro\cyrn\cyra\cyrv\cyre\cyrc\cyrsftsn}" }
970 ⟨utf8⟩             { "{виконавець}" }
971         { "language is not defined: " curlanguage " in bbl.executor" * * warning$ "executor" }
972       if$}
973     if$}
974 if$}
975
```

bbl.media

```
976 FUNCTION {bbl.media}
977 { media "online" =
978     { bbl.media.online }
```

```
979     { media "eresource" =
980       { bbl.media.eresource }
981       { bbl.media.text }
982     if$}
983 if$}
984
```

bbl.req

```
985 FUNCTION {bbl.req}
986 {
987   curlanguage "english" =
988     { "req." }
989     { curlanguage "german" =
990       { "ang." }
991       { curlanguage "russian" =
992 ⟨!utf8⟩        { "{\cyr\cyrz\cyra\cyrya\cyrv\cyrl.}" }
993 ⟨utf8⟩         { "{заявл.}" }
994         { "language is not defined: " curlanguage " in bbl.req" * * warning$ "req" }
995       if$
996       }
997     if$
998     }
999   if$
1000 }
1001
```

bbl.publ

```
1002 FUNCTION {bbl.publ}
1003 {
1004   curlanguage "english" =
1005     { "publ." }
1006     { curlanguage "german" =
1007       { "ausg." }
1008       { curlanguage "russian" =
1009 ⟨!utf8⟩        { "{\cyr\cyro\cyrp\cyru\cyrb\cyrl.}" }
1010 ⟨utf8⟩         { "{опубл.}" }
1011         { "language is not defined: " curlanguage " in bbl.publication" * * warning$ "publication" }
1012       if$
1013       }
1014     if$
1015     }
1016   if$
1017 }
1018
```

bbl.priority

```
1019 FUNCTION {bbl.priority}
1020 {
1021   curlanguage "english" =
1022     { "priority" }
1023     { curlanguage "german" =
1024       { "Prioritat" }
```

```
1025        { curlanguage "russian" =
1026 ⟨!utf8⟩         { "{\cyr\cyrp\cyrr\cyri\cyro\cyrr\cyri\cyrt\cyre\cyrt}" }
1027 ⟨utf8⟩        { "{приоритет}" }
1028          { "language is not defined: " curlanguage " in bbl.priority" * * warning$ "priority" }
1029        if$
1030        }
1031      if$
1032      }
1033    if$
1034 }
1035
```

New in version 1.2k.

```
1036 FUNCTION {bbl.jan}
1037 { curlanguage "english" =
1038   {"Jan."}
1039   { curlanguage "ukrainian" =
1040 ⟨!utf8⟩      {"\CYRS\cyrii\cyrch."}
1041 ⟨utf8⟩       {"Січ."} % Січень
1042        { curlanguage "russian" =
1043 ⟨!utf8⟩         { "\CYRYA\cyrn\cyrv." }
1044 ⟨utf8⟩         { "Янв." }
1045          { curlanguage "german" =
1046            { "Jan." } % Januar
1047            { "language is not defined: bbl.jan for " curlanguage * warning$ "Jan." }
1048          if$}
1049        if$}
1050    if$}
1051 if$}
1052
1053 FUNCTION {bbl.feb}
1054 { curlanguage "english" =
1055   {"Feb."}
1056   { curlanguage "ukrainian" =
1057 ⟨!utf8⟩      {"\CYRL\cyryu\cyrt."}
1058 ⟨utf8⟩      {"Лют."} % Лютий
1059      { curlanguage "russian" =
1060 ⟨!utf8⟩         { "\CYRF\cyre\cyrv\cyrr." }
1061 ⟨utf8⟩         { "Фев." }
1062          { curlanguage "german" =
1063            {"Feb."} % Februar
1064            {"language is not defined: bbl.feb for " curlanguage * warning$ "Feb."}
1065          if$}
1066        if$}
1067    if$}
1068 if$}
1069
1070 FUNCTION {bbl.mar}
1071 { curlanguage "english" =
1072   {"Mar."}
1073   { curlanguage "ukrainian" =
```

31

```
1074 ⟨!utf8⟩        {"\CYRB\cyre\cyrr."}
1075 ⟨utf8⟩        {"Бер."} % Березень
1076        { curlanguage "russian" =
1077 ⟨!utf8⟩              { "\CYRM\cyra\cyrr\cyrt" }
1078 ⟨utf8⟩              { "Март" }
1079          { curlanguage "german" =
1080            {"März"} % März
1081            {"language is not defined: bbl.mar for " curlanguage * warning$ "Mar."}
1082          if$}
1083        if$}
1084    if$}
1085 if$}
1086
1087 FUNCTION {bbl.apr}
1088 { curlanguage "english" =
1089    {"Apr."}
1090    { curlanguage "ukrainian" =
1091 ⟨!utf8⟩        {"\CYRK\cyrv\cyrii\cyrt."}
1092 ⟨utf8⟩        {"Квіт."} % квітень
1093        { curlanguage "russian" =
1094 ⟨!utf8⟩              { "\CYRA\cyrp\cyrr." }
1095 ⟨utf8⟩              { "Апр." }
1096          { curlanguage "german" =
1097            {"Apr."} % April
1098            { "language is not defined: bbl.apr for " curlanguage * warning$ "Apr." }
1099          if$}
1100        if$}
1101    if$}
1102 if$}
1103
1104 FUNCTION {bbl.may}
1105 { curlanguage "english" =
1106    {"May"}
1107    { curlanguage "ukrainian" =
1108 ⟨!utf8⟩        {"\CYRT\cyrr\cyra\cyrv."}
1109 ⟨utf8⟩        {"Трав."} % травень
1110        { curlanguage "russian" =
1111 ⟨!utf8⟩              { "\CYRM\cyra\cyrishrt" }
1112 ⟨utf8⟩              { "Май" }
1113          { curlanguage "german" =
1114            {"Mai"}
1115            { "language is not defined: bbl.may for " curlanguage * warning$ "May" }
1116          if$}
1117        if$}
1118    if$}
1119 if$}
1120
1121 FUNCTION {bbl.jun}
1122 { curlanguage "english" =
1123    {"June"}
```

```
1124     { curlanguage "ukrainian" =
1125 ⟨!utf8⟩         {"\CYRCH\cyre\cyrr."}
1126 ⟨utf8⟩          {"Чер."} % червень
1127         { curlanguage "russian" =
1128 ⟨!utf8⟩               { "\CYRI\cyryu\cyrn\cyrsftsn" }
1129 ⟨utf8⟩               { "Июнь" }
1130           { curlanguage "german" =
1131             {"Juni"}
1132             { "language is not defined: bbl.jun for " curlanguage * warning$ "June" }
1133           if$}
1134         if$}
1135     if$}
1136 if$}
1137
1138 FUNCTION {bbl.jul}
1139 { curlanguage "english" =
1140     {"July"}
1141     { curlanguage "ukrainian" =
1142 ⟨!utf8⟩         {"\CYRL\cyri\cyrp."}
1143 ⟨utf8⟩          {"Липень"} %Липень
1144         { curlanguage "russian" =
1145 ⟨!utf8⟩               { "\CYRI\cyryu\cyrl\cyrsftsn" }
1146 ⟨utf8⟩               { "Июль" }
1147           { curlanguage "german" =
1148             {"Juli"}
1149             { "language is not defined: bbl.jul for " curlanguage * warning$ "July" }
1150           if$}
1151         if$}
1152     if$}
1153 if$}
1154
1155 FUNCTION {bbl.aug}
1156 { curlanguage "english" =
1157     {"Aug."}
1158     { curlanguage "ukrainian" =
1159 ⟨!utf8⟩         {"\CYRS\cyre\cyrr."}
1160 ⟨utf8⟩          {"Серпень"} % Серпень
1161         { curlanguage "russian" =
1162 ⟨!utf8⟩               { "\CYRA\cyrv\cyrg\." }
1163 ⟨utf8⟩               { "Авг." }
1164           { curlanguage "german" =
1165             {"Aug."} % August
1166             { "language is not defined: bbl.aug for " curlanguage * warning$ "Aug." }
1167           if$}
1168         if$}
1169     if$}
1170 if$}
1171
1172 FUNCTION {bbl.sep}
1173 { curlanguage "english" =
```

```
1174     {"Sep."}
1175     { curlanguage "ukrainian" =
1176 ⟨!utf8⟩      {"\CYRV\cyre\cyrr."}
1177 ⟨utf8⟩       {"Вер."} % вересень
1178        { curlanguage "russian" =
1179 ⟨!utf8⟩           { "\CYRS\cyre\cyrn\cyrt." }
1180 ⟨utf8⟩            { "Сент." }
1181           { curlanguage "german" =
1182              {"Sep."} % September
1183              { "language is not defined: bbl.sep for " curlanguage * warning$ "Sep." }
1184           if$}
1185        if$}
1186     if$}
1187 if$}
1188
1189 FUNCTION {bbl.oct}
1190 { curlanguage "english" =
1191     {"Oct."}
1192     { curlanguage "ukrainian" =
1193 ⟨!utf8⟩      {"\CYRZH\cyro\cyrn."}
1194 ⟨utf8⟩       {"Жов."} % жовтень
1195        { curlanguage "russian" =
1196 ⟨!utf8⟩           { "\CYRO\cyrk\cyrt." }
1197 ⟨utf8⟩            { "Окт." }
1198           { curlanguage "german" =
1199              {"Okt."} % Oktober
1200              { "language is not defined: bbl.oct for " curlanguage * warning$ "Oct." }
1201           if$}
1202        if$}
1203     if$}
1204 if$}
1205
1206 FUNCTION {bbl.nov}
1207 { curlanguage "english" =
1208     {"Nov."}
1209     { curlanguage "ukrainian" =
1210 ⟨!utf8⟩      {"\CYRL\cyri\cyrs."}
1211 ⟨utf8⟩       {"Лис."} % листопад
1212        { curlanguage "russian" =
1213 ⟨!utf8⟩           { "\CYRN\cyro\cyrya\cyrb." }
1214 ⟨utf8⟩            { "Нояб." }
1215           { curlanguage "german" =
1216              {"Nov."} % November
1217              { "language is not defined: bbl.nov for " curlanguage * warning$ "Nov." }
1218           if$}
1219        if$}
1220     if$}
1221 if$}
1222
1223 FUNCTION {bbl.dec}
```

```
1224 { curlanguage "english" =
1225     {"Dec."}
1226     { curlanguage "ukrainian" =
1227 ⟨!utf8⟩      {"\CYRG\cyrr\cyru\cyrd."}
1228 ⟨utf8⟩       {"Груд."} % грудень
1229         { curlanguage "russian" =
1230 ⟨!utf8⟩           { "\CYRD\cyre\cyrk." }
1231 ⟨utf8⟩            { "Дек." }
1232           { curlanguage "german" =
1233             {"Dez."} % Dezember
1234             { "language is not defined: bbl.dec for " curlanguage * warning$ "Dec." }
1235           if$}
1236       if$}
1237     if$}
1238 if$}
```

```
1239 FUNCTION {bbl.arxiv}
1240 { curlanguage "english" =
1241     { "ArXiv" }
1242     { curlanguage "ukrainian" =
1243 ⟨!utf8⟩      {"ArXiv"}
1244 ⟨utf8⟩       {"ArXiv"}
1245         { curlanguage "russian" =
1246 ⟨!utf8⟩          { "ArXiv" }
1247 ⟨utf8⟩           { "ArXiv" }
1248           { curlanguage "german" =
1249             { "ArXiv" }
1250             { "language is not defined: bbl.arxiv for " curlanguage * warning$ "ArXiv" }
1251           if$}
1252       if$}
1253     if$}
1254 if$}
1255
1256 FUNCTION {bbl.jstor}
1257 { curlanguage "english" =
1258     { "JSTOR" }
1259     { curlanguage "ukrainian" =
1260 ⟨!utf8⟩      {"JSTOR"}
1261 ⟨utf8⟩       {"JSTOR"}
1262         { curlanguage "russian" =
1263 ⟨!utf8⟩          { "JSTOR" }
1264 ⟨utf8⟩           { "JSTOR" }
1265           { curlanguage "german" =
1266             { "JSTOR" }
1267             { "language is not defined: bbl.jstor for " curlanguage * warning$ "JSTOR" }
1268           if$}
1269       if$}
1270     if$}
1271 if$}
1272
```

```
1273 FUNCTION {bbl.pubmed}
1274 { curlanguage "english" =
1275     { "PubMed" }
1276     { curlanguage "ukrainian" =
1277 ⟨!utf8⟩       {"PubMed"}
1278 ⟨utf8⟩        {"PubMed"}
1279       { curlanguage "russian" =
1280 ⟨!utf8⟩           { "PubMed" }
1281 ⟨utf8⟩            { "PubMed" }
1282         { curlanguage "german" =
1283           { "PubMed" }
1284           { "language is not defined: bbl.pubmed for " curlanguage * warning$ "PubMed" }
1285         if$}
1286       if$}
1287     if$}
1288 if$}
1289
1290 FUNCTION {bbl.googlebooks}
1291 { curlanguage "english" =
1292     { "Google Books" }
1293     { curlanguage "ukrainian" =
1294 ⟨!utf8⟩       {"Google \CYRK\cyrn\cyri\cyrg\cyri"}
1295 ⟨utf8⟩        {"Google Книги"}
1296       { curlanguage "russian" =
1297 ⟨!utf8⟩           { "Google \CYRK\cyrn\cyri\cyrg\cyri" }
1298 ⟨utf8⟩            { "Google Книги" }
1299         { curlanguage "german" =
1300           { "Google Books" }
1301           { "language is not defined: bbl.googlebooks for " curlanguage * warning$ "Google Books" }
1302         if$}
1303       if$}
1304     if$}
1305 if$}
1306
1307 FUNCTION {bbl.hdl}
1308 { curlanguage "english" =
1309     { "Handle.Net" }
1310     { curlanguage "ukrainian" =
1311 ⟨!utf8⟩       {"Handle.Net"}
1312 ⟨utf8⟩        {"Handle.Net"}
1313       { curlanguage "russian" =
1314 ⟨!utf8⟩           { "Handle.Net" }
1315 ⟨utf8⟩            { "Handle.Net" }
1316         { curlanguage "german" =
1317           { "Handle.Net" }
1318           { "language is not defined: bbl.hdl for " curlanguage * warning$ "Handle.Net" }
1319         if$}
1320       if$}
1321     if$}
1322 if$}
```

6.6 Aliases

Aliases to some fields are introduced with the help of *field1* .or. *field2* functions.

address.or.location   Pushes `address` field if not empty; otherwise pushes `location` value even if it is empty.

```
1323 %FUNCTION {address.or.location}
1324 %{
1325 %  address empty$
1326 %    { location empty$
1327 %      'skip$
1328 %      { location }
1329 %    if$ }
1330 %    { address }
1331 %  if$
1332 %}
1333 FUNCTION {address.or.location}
1334 {
1335   address empty$
1336     { location }
1337     { address }
1338   if$
1339 }
1340
```

itycode.or.number   Pushed `specialitycode` value if not empty; otherwise returns `number` value even if the latter is empty.

```
1341 FUNCTION {specialitycode.or.number}
1342 {
1343   specialitycode empty$
1344     { number }
1345     { specialitycode }
1346   if$
1347 }
1348
```

itution.or.school   Pushes `institution` value if not empty; otherwise returns `school` value even if the latter is empty.

```
1349 FUNCTION {institution.or.school}
1350 {
1351   institution empty$
1352     { school}
1353     { institution }
1354   if$
1355 }
```

6.7 Formatting dates

⟨NB: NEEDS to be located before natbib labels. This is experimental section. Needs to be upgraded.⟩

date.to.year   New in version 1.2k. Extracts year from date. Currently, only the date of the form either `YYYY-MM-DD`, or `YYYY-MM`, or `YYYY` can be processed as expected. Returns `date` if it is empty or has wrong format.

```
1356 FUNCTION {date.to.year}
1357 {
1358   date empty$
1359     { date }
1360     {
1361 ⟨∗debug⟩
1362         "date.to.year::: date in " cite$ * " =" * date * warning$
1363         "date.to.year::: length of date is " date text.length$ int.to.str$ * warning$
1364 ⟨/debug⟩
1365         date text.length$ #3 >
1366           {
1367             date #1 #4 substring$ 'y :=
1368 ⟨∗debug⟩
1369             ":::::: y=" y * warning$
1370 ⟨/debug⟩
1371             y
1372           }
1373           {
1374             "wrong format of date in " cite$ * ": date=" * date * warning$
1375             date
1376           }
1377         if$
1378     }
1379   if$
1380 }
1381
```

date.to.month   New in version 1.2k. Extracts month from date. Returns `date` if it is empty or has wrong format.

```
1382 FUNCTION {date.to.month}
1383 {
1384   date empty$
1385     { date }
1386     {
1387 ⟨∗debug⟩
1388         "date.to.month::: date in " cite$ * " =" * date * warning$
1389         "date.to.month::: length of date is " date text.length$ int.to.str$ * warning$
1390 ⟨/debug⟩
1391         date text.length$ #6 >
1392           {
1393             date #6 #2 substring$ 'm :=
1394 ⟨∗debug⟩
1395             ":::::: m=" m * warning$
1396 ⟨/debug⟩
1397             m
1398           }
1399           {
1400             "wrong format of date in " cite$ * ": date=" * date * warning$
1401             date
1402           }
1403         if$
```

```
1404       }
1405    if$
1406 }
1407
```

date.to.day   New in version 1.2k. Extracts year from date. Returns `date` if it is empty or has wrong format.

```
1408 FUNCTION {date.to.day}
1409 {
1410   date empty$
1411     { date }
1412     {
1413 ⟨∗debug⟩
1414       "date.to.day::: date in " cite$ * " =" * date * warning$
1415       "date.to.day::: length of date is " date text.length$ int.to.str$ * warning$
1416 ⟨/debug⟩
1417       date text.length$ #10 =
1418         {
1419           date #9 #2 substring$ 'd :=
1420 ⟨∗debug⟩
1421           ":::::: d=" d * warning$
1422 ⟨/debug⟩
1423           d
1424         }
1425         {
1426           "wrong format of date in " cite$ * ": date=" * date * warning$
1427           date
1428         }
1429       if$
1430     }
1431   if$
1432 }
1433
```

r.or.date.to.year   New in version 1.2k. Returns `year` if not empty; otherwise call `date.to.year`.
⟨№: ВОЗМОЖНО, ПОДОБНЫЕ ФУНКЦИИ ЛУЧШЕ назвать `this.year`.⟩

```
1434 FUNCTION {year.or.date.to.year}
1435 {
1436   year empty$
1437     {
1438 ⟨∗debug⟩
1439       "year.or.date.to.year::: empty year in " cite$ * warning$
1440 ⟨/debug⟩
1441       date.to.year
1442     }
1443     { year }
1444   if$
1445 }
1446
```

format.month   Reads `month` field and translate standard English abbreviation of months (as defined by

jan, feb, etc. macros below) to current language. ⟨NB: НАДО ИЗОБРЕСТИ ЧТО-ТО ВРОДЕ
month.or.date.to.month (this.month).⟩

```
1447 FUNCTION {format.month}
1448 { month empty$
1449     { "" }
1450     { month "Jan." =
1451         { bbl.jan }
1452     { month "Feb." =
1453         { bbl.feb }
1454     { month "Mar." =
1455         { bbl.mar }
1456     { month "Apr." =
1457         { bbl.apr }
1458     { month "May" =
1459         { bbl.may }
1460     { month "Jun." =
1461         { bbl.jun }
1462     { month "Jul." =
1463         { bbl.jul }
1464     { month "Aug." =
1465         { bbl.aug }
1466     { month "Sep." =
1467         { bbl.sep }
1468     { month "Oct." =
1469         { "Окт." }
1470     { month "Nov." =
1471         { bbl.nov }
1472     { month "Dec." =
1473         { bbl.dec }
1474         {
1475           "unknown month in " cite$ * warning$
1476           month
1477         }
1478         if$}if$}if$}if$}if$}if$}if$}if$}if$}if$}if$}if$}
1479     if$
1480 }
```

```
1481 FUNCTION {format.date}
1482 %{ year empty$
1483 { year.or.date.to.year empty$
1484     { month empty$
1485         { "" }
1486         { "there's a month but no year in " cite$ * warning$
1487           format.month
1488         }
1489       if$
1490     }
1491     { month empty$
1492 %       'year
```

40

```
1493 %              { year ". \BibDash " format.month * * }
1494          'year.or.date.to.year
1495          { year.or.date.to.year ". \BibDash " format.month * * }
1496       if$
1497     }
1498   if$
1499 }
1500
```

6.8 Formatting names

Declare functions to format separate elements of a bibliographic reference.

Important note

Neither `bibtex` nor `bibtex8` can handle unicoded text without troubles. In particular, they fail to reduce a Cyrillic name to initials. Therefore we avoid using `f.` primitive (which trims first name to first letter) when option `utf8` is in effect; in the latter caase `ff` primitive is called instead.

```
1501 INTEGERS { nameptr namesleft numnames }
1502
```

<div style="margin-left:0">fmt.names.first
fmt.names.three
fmt.names.all</div>

New version of functions formating names. Function name indicates number of persons printed. These functions look for last literal in the stack which should not be empty; hence, they should be called by other functions that checks if the last literal is empty.

```
1503 FUNCTION {fmt.names.first}
1504 { #1
1505 ⟨strict&!utf8⟩   "{vv~}{ll}{~jj}{,~f.}"
1506 ⟨strict & utf8⟩   "{vv~}{ll}{~jj}{,~ff}"
1507 ⟨!strict&!utf8⟩   "{vv~}{ll}{~jj}{~f.}"
1508 ⟨!strict & utf8⟩   "{vv~}{ll}{~jj}{~ff}"
1509   format.name$
1510 }
1511
1512 FUNCTION {fmt.names.three}
1513 {
1514   's :=
1515   #1 'nameptr :=
1516   s num.names$ 'numnames :=
1517   numnames 'namesleft :=
1518   { namesleft #0 > }
1519   { s nameptr
1520 ⟨strict&!utf8⟩        "{vv~}{ll}{~jj}{,~f.}"
1521 ⟨strict & utf8⟩        "{vv~}{ll}{~jj}{,~ff}"
1522 ⟨!strict&!utf8⟩        "{vv~}{ll}{~jj}{~f.}"
1523 ⟨!strict & utf8⟩        "{vv~}{ll}{~jj}{~ff}"
1524     format.name$ 't :=
1525     nameptr #1 >
1526       { nameptr #4 = numnames #4 > and
1527           { "others" 't :=
1528             #1 'namesleft :=
1529           }
```

41

```
1530            'skip$
1531          if$
1532          namesleft #1 >
1533            { ", " * t * }
1534            { t "others" = t "~others" = or
1535 ⟨!strict⟩            { " " * bbl.etal * }
1536 ⟨strict⟩             { " " * bbl.etal bracketise *}
1537              { ", " * t * }
1538            if$
1539            }
1540          if$
1541        }
1542        't
1543    if$
1544    nameptr #1 + 'nameptr :=
1545    namesleft #1 - 'namesleft :=
1546  }
1547  while$
1548 }
1549
1550 FUNCTION {fmt.names.all}
1551 { 's :=
1552   #1 'nameptr :=
1553   s num.names$ 'numnames :=
1554   numnames 'namesleft :=
1555     { namesleft #0 > }
1556     { s nameptr
1557       %"{vv~}{ll}" format.name$ 't :=
1558 ⟨strict&!utf8⟩       "{vv~}{ll}{~jj}{,~f.}"
1559 ⟨strict & utf8⟩      "{vv~}{ll}{~jj}{,~ff}"
1560 ⟨!strict&!utf8⟩      "{vv~}{ll}{~jj}{~f.}"
1561 ⟨!strict & utf8⟩     "{vv~}{ll}{~jj}{~ff}"
1562       format.name$ 't :=
1563       nameptr #1 >
1564         { namesleft #1 >
1565            { ", " * t * }
1566            { numnames #2 >  curlanguage "english" = and
1567               { "," * }
1568               'skip$
1569             if$
1570             t "others" = t "~others" = or
1571 ⟨!strict⟩              { " " * bbl.etal * }
1572 ⟨strict⟩              { " " * bbl.etal bracketise *}
1573             { " " bbl.and " " * * * t * }
1574             if$
1575           }
1576         if$
1577       }
1578       't
1579     if$
```

```
1580        nameptr #1 + 'nameptr :=
1581        namesleft #1 - 'namesleft :=
1582     }
1583   while$
1584 }
1585
```

6.9 Formatting names (cont.)

format.author   Formats a list of authors for the heading part of a bibliographic record by appying either
`fmt.names.first`, `fmt.names.three` or `fmt.names.all` to the field `author`, if it is not empty;
otherwise it pushes empty `author`. This allows making a check as in the case of `format.author "author" output.check`.

⟨NB: Однако заметим, что все другие функции `format....` пустое поле замещают пустой
строкой.⟩

```
1586 %FUNCTION {format.author}
1587 %{ author empty$
1588 %    %%%{ "" }  % < v.1.2k
1589 %    { author } % v.1.2k
1590 %    %{ author format.names emphasize} % 1st if strict, <=3 otherwise
1591 %%<long>    { author fmt.names.all emphasize}
1592 %%<!long&strict>    { author fmt.names.first emphasize}
1593 %%<!long&!strict>    { author fmt.names.three emphasize}
1594 %  if$
1595 %}
1596 FUNCTION {format.author}
1597 {
1598   author empty$
1599     { author } %%%'skip$
1600 ⟨*long⟩
1601     { author fmt.names.all }
1602 ⟨/long⟩
1603 ⟨*!long⟩
1604    {author num.names$ #4 <
1605       {
1606 ⟨strict⟩         author fmt.names.first
1607 ⟨!strict⟩          author fmt.names.three
1608       }
1609       { "" } %%%'skip$
1610     if$}
1611 ⟨/!long⟩
1612   if$
1613 }
1614
```

format.bookauthors   Is used only once by `bookauthor.head` called in `inbook` entry. ⟨NB: Not used anymore!⟩

```
1615 ⟨*debug⟩
1616 %FUNCTION {format.bookauthors}
1617 %{ bookauthor empty$
1618 %    { "" }
1619 %    { bookauthor format.names}% cuts to 4 persons if |!strict| option
```

43

```
1620 %  if$
1621 %}
1622 %
1623 ⟨/debug⟩
```

format.author.rest   Formats `author` to be placed after a slash in the zone of responsibiloty of a bibliographic record. In contrast to `output.author.rest` does not check number of items in the `author` field; it is recommended to use `output.author.rest` instead whenever possible. ⟨NB: Not used any more.⟩

```
1624 ⟨*debug⟩
1625 FUNCTION {format.author.rest}
1626 {
1627 ⟨*long⟩
1628   %%% Does this work?
1629   %%%skip$    % this seemed to work
1630   %%% ""      % this seemed to work
1631 ⟨/long⟩
1632 ⟨*!long⟩
1633   author empty$
1634     %%%{ "" } % < v.1.2k
1635     { author } % v.1.2k
1636     { author fmt.names.three }
1637   if$
1638 ⟨/!long⟩
1639 }
1640
1641 ⟨/debug⟩
```

.bookauthors.rest

```
1642 FUNCTION {format.bookauthors.rest}
1643 { bookauthor empty$
1644     { "" }
1645 ⟨long⟩    { bookauthor fmt.names.all emphasize }
1646 ⟨!long⟩    { bookauthor fmt.names.three emphasize }
1647   if$
1648 }
1649
```

rmat.editors.rest

```
1650 FUNCTION {format.editors.rest}
1651 { editor empty$
1652     { "" }
1653 ⟨long⟩    { bbl.edby "\ " * editor fmt.names.all * }
1654 ⟨!long⟩    { bbl.edby "\ " * editor fmt.names.three * }
1655   if$
1656 }
1657
```

format.chief.rest   Formats `editor` field for `report` and `techreport` entries.

```
1658 FUNCTION {format.chief.rest}
1659 { editor empty$
1660     { "" }
1661 ⟨long⟩    { bbl.chief "\ " * editor fmt.names.all * }
```

1662 ⟨!long⟩   { bbl.chief "\ " * editor fmt.names.three * }
1663   if$
1664 }
1665

1666 FUNCTION {format.executor.rest}
1667 { author empty$
1668   { "" }
1669 ⟨long⟩   { bbl.executor ": " * author fmt.names.all * }
1670 ⟨!long⟩   { bbl.executor ": " * author fmt.names.three * }
1671   if$
1672 }
1673

1674 FUNCTION {format.compiler.rest}
1675 { compiler empty$
1676   { "" }
1677 ⟨long⟩   { bbl.compiler "\ " * compiler fmt.names.all * }
1678 ⟨!long⟩   { bbl.compiler "\ " * compiler fmt.names.three * }
1679   if$
1680 }
1681

6.10   Formatting natbib keys

1682 ⟨∗natbib⟩

fmt.names.brief   Formats all names like fmt.names.all except that first name of every person is dropped (dropped ff and f. modifiers).

⟨NB: Вероятно, нужно сократить число персон до 3х максимум. Что говорит ГОСТ? Где используется это? В метке ссылки это расгиренный список авторов после краткого списка и года в необязательном аргументе \bibitem.⟩

1683 FUNCTION {fmt.names.brief}
1684 { 's :=
1685   #1 'nameptr :=
1686   s num.names$ 'numnames :=
1687   numnames 'namesleft :=
1688   { namesleft #0 > }
1689   { s nameptr "{vv~}{ll}" format.name$ 't :=
1690   nameptr #1 >
1691   {
1692   namesleft #1 >
1693   { ", " * t * }
1694   {
1695   numnames #2 > curlanguage "english" = and
1696   { "," * }
1697   'skip$
1698   if$
1699   t "others" = t "~others" = or

```
1700                  { " " bbl.etal * * }
1701                    { " " bbl.and " " * * * t * }
1702                  if$
1703              }
1704           if$
1705        }
1706        't
1707      if$
1708      nameptr #1 + 'nameptr :=
1709      namesleft #1 - 'namesleft :=
1710    }
1711  while$
1712 }
1713
```

format.names.key Declares function to go to left part of optional argument of \bibitem in the bibstyles generated
with the option natbib. It cuts the list of person to 2 at most and drops first name of every person.

```
1714 FUNCTION {format.names.key}
1715 { 's :=
1716   set.language %%% уже вызвана в output.bibitem, но вроде бы нужна и здесь
1717   s #1 "{vv~}{ll}" format.name$
1718   s num.names$ duplicate$
1719   #2 >
1720     { pop$ " " bbl.etal * * }
1721     { #2 <
1722       'skip$
1723       { s #2 "{ff }{vv }{ll}{ jj}" format.name$ "others" =
1724           { " " bbl.etal * * }
1725           { " " bbl.and " " * * * s #2 "{vv~}{ll}" format.name$ * }
1726         if$
1727       }
1728     if$
1729     }
1730   if$
1731 }
1732
```

format.key Substitute an empty last literal in the stack (usually, author) with the key field if provided.
Used as a heading of bibliographic record of author is empty.

```
1733 FUNCTION {format.key}
1734 { empty$
1735     { key field.or.null }
1736     { "" }
1737   if$
1738 }
1739
```

author.key.label Composes a key to be used as a reference label with natbib styles. If author field is empty an
attempt is made to retrieve key field. If it is also empty 3 first letters are retreived from the citation
key cite$. If author field is not empty format.names.key defined above is called.

```
1740 FUNCTION {author.key.label}
1741 { author empty$
```

```
1742      { key empty$
1743          { cite$ #1 #3 substring$ }
1744          'key
1745        if$
1746      }
1747      { author format.names.key }
1748    if$
1749 }
```

.editor.key.label   These functions operate similarly but engage a different set of fields. They are called below
y.organization.label by calc.short.list.
y.organization.label

```
1750 FUNCTION {author.editor.key.label}
1751 { author empty$
1752      { editor empty$
1753          { key empty$
1754              { cite$ #1 #3 substring$ }
1755              %'key              %% causes lost of year
1756              { "{}" key * } %% Bug in bibtex8 ??
1757            if$
1758          }
1759          { editor format.names.key }
1760        if$
1761      }
1762      { author format.names.key }
1763    if$
1764 }
1765
1766 FUNCTION {author.key.organization.label}
1767 { author empty$
1768      { key empty$
1769          { organization empty$
1770              { cite$ #1 #3 substring$ }
1771              { "The " #4 organization chop.word #3 text.prefix$ }
1772            if$
1773          }
1774          'key
1775        if$
1776      }
1777      { author format.names.key }
1778    if$
1779 }
1780
1781 FUNCTION {editor.key.organization.label}
1782 { editor empty$
1783      { key empty$
1784          { organization empty$
1785              { cite$ #1 #3 substring$ }
1786              { "The " #4 organization chop.word #3 text.prefix$ }
1787            if$
1788          }
1789          'key
```

```
1790        if$
1791      }
1792      { editor format.names.key }
1793    if$
1794 }
1795
```

calc.short.list   Calculates `short.list` for `\bibitem` label in natbib styles depending on the type of entry.

⟨NB: Нужна внимательная ревизия логики вычислений. В общем-то, она имеет значения в особом случае, когда поле author пустое. ⟩

```
1796 FUNCTION {calc.short.list}
1797 { type$ "book" = type$ "inbook" = or
1798     'author.editor.key.label
1799     { type$ "proceedings" =
1800         'editor.key.organization.label
1801         { type$ "manual" =
1802             'author.key.organization.label
1803             'author.key.label
1804           if$
1805         }
1806       if$
1807     }
1808   if$
1809   'short.list :=
1810 }
1811
```

calc.label

⟨NB: Вопреки названию, вычисляет только часть метки в [] для `\bibitem`. Скобка после года не закрыта, так как еще может быть добавлено a, b, ... Лучше перенести весь код output.bibitem. ⟩

```
1812 FUNCTION {calc.label}
1813 { calc.short.list
1814   short.list
1815   "("
1816   *
1817 %  year duplicate$ empty$
1818   year.or.date.to.year duplicate$ empty$
1819   short.list key field.or.null = or
1820     { pop$ "" }
1821     'skip$
1822   if$
1823   *
1824   'label :=
1825 }
1826
```

calc.long.list   In case of `natbib` option, we need `calc.long.list` to compose `output.bibitem`, and the latter, in its turn, requires some more functions.

```
1827 FUNCTION {calc.long.list} %% called 1 time only
1828 { type$ "book" = type$ "inbook" = or
1829     %'format.author.editor.brief
```

```
1830     { author empty$
1831         { editor empty$
1832             { "" }
1833             { editor fmt.names.brief }
1834           if$
1835         }
1836         { author fmt.names.brief }
1837       if$
1838     }
1839     { type$ "proceedings" =
1840         %'format.editor.brief
1841         { editor empty$
1842             { "" }
1843             { editor fmt.names.brief }
1844           if$
1845         }
1846         %'format.author.brief
1847         { author empty$
1848             { "" }
1849             { author fmt.names.brief }
1850           if$
1851         }
1852       if$
1853     }
1854   if$
1855 }
1856
1857 ⟨/natbib⟩
```

6.11   Output functions (continued)

address.publisher   Outputs address (or location) and publisher fields separated by colon if both fields are available; otherwise outputs that field which is not empty.

```
1858 ⟨∗!strict⟩
1859 FUNCTION {output.address.publisher}
1860 {
1861   address empty$ location empty$ and
1862     'skip$
1863     { address.or.location output
1864       publisher empty$
1865         'skip$
1866         { new.colon }
1867       if$
1868     }
1869   if$
1870   publisher output
1871 }
1872 ⟨/!strict⟩
1873 ⟨∗strict⟩
1874 FUNCTION {output.address.publisher}
```

49

```
1875 {
1876   address empty$
1877     {
1878       bbl.nnoaddress
1879       publisher empty$
1880         { "~: " * bbl.nopublisher * bracketise }
1881         { bracketise "~: " * publisher * }
1882       if$
1883     }
1884     {
1885       address output
1886       new.colon
1887       publisher empty$
1888         { bbl.nopublisher bracketise }
1889         { publisher }
1890       if$
1891     }
1892   if$
1893   output
1894 }
1895
1896 ⟨/strict⟩
```

output.bibitem   Is called at the beginning of any entry. It sets `curlanguage` string variable based on `langid` or `language` field.

```
1897 FUNCTION {output.bibitem}
1898 {
1899   set.language
1900   newline$
1901   "\bibitem" write$
1902 ⟨∗natbib⟩
1903   label extra.label * ")" *
1904   calc.long.list *
1905   bracketise write$
1906 ⟨/natbib⟩
1907   cite$ bracify write$
1908   newline$
1909   "\selectlanguageifdefined" curlanguage bracify * write$
1910   newline$
1911   ""
1912   before.all 'output.state :=
1913 }
1914
1915 ⟨∗natbib⟩
1916 %FUNCTION {output.bibitem}
1917 %{ newline$
1918 %  "\bibitem" write$
1919 %%  author.key.label
1920 %%  year parenthesify *
1921 %%  "; lbl:" label * *
```

50

```
1922 %%   "; mfn:"  calc.long.list * *
1923 %   label
1924 %   calc.long.list *
1925 %   bracketise write$
1926 %   cite$ bracify write$
1927 %   newline$
1928 %   language empty$
1929 %     { "english" 'curlanguage := }
1930 %     {language   'curlanguage := }
1931 %   if$
1932 %   "\selectlanguageifdefined" curlanguage bracify * write$
1933 %   newline$
1934 %   ""
1935 %   before.all 'output.state :=
1936 %}
1937
1938 ⟨/natbib⟩
```

6.12   Formatting title, booktitle, etc.

Important note

Neither `bibtex.exe` nor `bibtex8.exe` can handle unicoded text without troubles. In particular, `bibtex8` fails to change case of a string if it contains Cyrillic letter. Therefore we avoid using `change.case$` when option `utf8` is applied.

format.bvolume

```
1939 FUNCTION {format.bvolume}
1940 { volume empty$
1941     { "" }
1942     { bbl.vvol volume tie.connect
1943       series empty$
1944         'skip$
1945         { bbl.of spaces.around * series emphasize * }
1946       if$
1947       "volume and number" number either.or.check
1948     }
1949   if$
1950 }
1951
```

format.number.series

```
1952 FUNCTION {format.number.series}
1953 { volume empty$
1954     { number empty$
1955         { series field.or.null }
1956         { series empty$
1957             { "there's a number but no series in " cite$ * warning$
1958               bbl.nnr }
1959             {
1960                 %new.dblslash
1961                 new.sentence
```

```
1962                 series
1963                 bbl.nr
1964                 tie.or.space.connect}
1965            if$
1966           number tie.or.space.connect
1967         }
1968      if$
1969    }
1970    { "" }
1971  if$
1972 }
1973
```

eng.ord   Is not currently used. ⟨NB: Note that `bbl.st`, `bbl.nd`, `bbl.rd`, `bbl.th` are not defined.⟩

```
1974 ⟨∗debug⟩
1975 FUNCTION {eng.ord}
1976 { duplicate$ "1" swap$ *
1977   #-2 #1 substring$ "1" =
1978     { bbl.th * }
1979     { duplicate$ #-1 #1 substring$
1980       duplicate$ "1" =
1981         { pop$ bbl.st * }
1982         { duplicate$ "2" =
1983             { pop$ bbl.nd * }
1984             { "3" =
1985                 { bbl.rd * }
1986                 { bbl.th * }
1987               if$
1988             }
1989           if$
1990         }
1991       if$
1992     }
1993   if$
1994 }
1995 ⟨/debug⟩
1996
```

convert.edition

```
1997 FUNCTION {convert.edition}
1998 { edition
1999 % edition extract.num "l" change.case$ 's :=
2000 %   s "first" = s "1" = or
2001 %     { bbl.first 't := }
2002 %     { s "second" = s "2" = or
2003 %         { bbl.second 't := }
2004 %         { s "third" = s "3" = or
2005 %             { bbl.third 't := }
2006 %             { s "fourth" = s "4" = or
2007 %                 { bbl.fourth 't := }
2008 %                 { s "fifth" = s "5" = or
```

```
2009 %                         { bbl.fifth 't := }
2010 %                         { s #1 #1 substring$ is.num
2011 %                             { s eng.ord 't := }
2012 %                             { edition 't := }
2013 %                           if$
2014 %                         }
2015 %                       if$
2016 %                     }
2017 %                   if$
2018 %                 }
2019 %               if$
2020 %             }
2021 %           if$
2022 %     }
2023 %   if$
2024 %   t
2025 }
2026
```

```
2027 FUNCTION {format.edition}
2028 { edition empty$
2029     { "" }
2030     { output.state mid.sentence =
2031 ⟨!utf8⟩        { convert.edition "l" change.case$ " " * bbl.edition * }
2032 ⟨!utf8⟩        { convert.edition "t" change.case$ " " * bbl.edition * }
2033 ⟨utf8⟩         { convert.edition " " * bbl.edition * }
2034 ⟨utf8⟩         { convert.edition " " * bbl.edition * }
2035       if$
2036     }
2037   if$
2038 }
2039
```

```
2040 FUNCTION {format.pages}
2041 { eid empty$
2042     {
2043       pages empty$
2044         { "" }
2045         { pages multi.page.check
2046             { bbl.ppages pages n.dashify tie.connect }
2047             { bbl.ppage pages tie.connect }
2048           if$
2049         }
2050       if$
2051     }
2052     { eid multi.page.check
2053         { bbl.ppages eid n.dashify tie.connect }
2054         { bbl.ppage eid tie.connect }
2055       if$
```

```
2056        }
2057    if$
2058 }
2059
```

```
2060 FUNCTION {format.pages.page}
2061 { eid empty$
2062     { pages empty$
2063         { pagetotal empty$
2064             { "" }
2065             { pagetotal bbl.pages tie.connect }
2066           if$
2067         }
2068         { format.pages}
2069       if$
2070     }
2071     { format.pages }
2072    if$
2073 }
2074
```

```
2075 FUNCTION {format.vol.num.pages}
2076 { volume field.or.null
2077   number empty$
2078     'skip$
2079     {
2080       ", no." number tie.or.space.connect *
2081       volume empty$
2082         { "there's a number but no volume in " cite$ * warning$ }
2083         'skip$
2084       if$
2085     }
2086   if$
2087   pages empty$
2088     'skip$
2089     { duplicate$ empty$
2090         { pop$ format.pages }
2091         { ": " * pages n.dashify * }
2092       if$
2093     }
2094   if$
2095 }
2096
```

```
2097 FUNCTION {format.volume}
2098 { volume empty$
2099     { "" }
2100     { bbl.vvol volume tie.or.space.connect }
2101    if$
```

54

```
2102 }
2103
```

```
2104 FUNCTION {format.number}
2105 { number empty$
2106     { "" }
2107     { bbl.nr number tie.or.space.connect }
2108   if$
2109 }
2110
```

```
2111 ⟨∗debug⟩
2112 FUNCTION {format.chapter.pages}
2113 { chapter empty$
2114     'format.pages
2115     { type empty$
2116         { bbl.chapter }
2117         { type "l" change.case$ }
2118       if$
2119       chapter tie.or.space.connect
2120       pages empty$
2121         'skip$
2122         { ", " * format.pages * }
2123       if$
2124     }
2125   if$
2126 }
2127 ⟨/debug⟩
2128
```

```
2129 FUNCTION {empty.misc.check}
2130 { author empty$ title empty$ howpublished empty$
2131   month empty$ year empty$ note empty$
2132   and and and and and
2133   key empty$ not and
2134     { "all relevant fields are empty in " cite$ * warning$ }
2135     'skip$
2136   if$
2137 }
2138
```

⟨NB: ЭТУ ФУНКЦИЮ ПЕРЕНЕСТИ КУДА-ТО ВНИЗ ближе к format.thesis.type И объединить с ней (?)⟩

```
2139 FUNCTION {bbl.thesis.type}
2140 { type "mathesis" =
2141   { bbl.mathesis }
2142   { type "phdthesis" =
2143     { bbl.phdthesis }
```

55

```
2144       { type "docthesis" =
2145          { bbl.docthesis }
2146          %%{ "!!!" type * "t" change.case$ }
2147 ⟨!utf8⟩       { type "t" change.case$ }
2148 ⟨utf8⟩        { type }
2149       if$}
2150    if$}
2151 if$}
2152
```

ormat.thesis.type

```
2153 %FUNCTION {format.thesis.type}
2154 %{ type empty$
2155 %     'skip$
2156 %     { pop$
2157 %%<!utf8>       bbl.thesis.type "t" change.case$
2158 %%<utf8>        bbl.thesis.type
2159 %     }
2160 %   if$
2161 %}
2162 FUNCTION {format.thesis.type}
2163 { type empty$
2164     'skip$
2165     { pop$
2166        bbl.thesis.type
2167     }
2168   if$
2169 }
```

chrep.type.number   Function to format report type and number.

```
2170 %FUNCTION {format.techrep.type.number}
2171 %{ type empty$
2172 %     { bbl.techreport }
2173 %     'type
2174 %   if$
2175 %   number empty$
2176 %%<!utf8>     { "t" change.case$ }
2177 %%<utf8>      { "" }
2178 %     { number tie.or.space.connect }
2179 %   if$
2180 %}
2181
2182 FUNCTION {format.techreport.type}
2183 { type empty$
2184     { bbl.techreport }
2185     'type
2186   if$
2187 }
2188
```

utput.author.head   Formats and writes list of authors in the heading of a bibliographic record. The GOST bibstyles
                         skip the list of authors in the beginning of the bibliographic record if the number of authors is 4

or larger except for the styles with suffix l which prints all authors in the heading. The bibstyles compiled with the option struct print at most 1 person name in the heading.

```
2189 FUNCTION {output.author.head}
2190 {
2191   author empty$
2192     'skip$
2193 ⟨∗long⟩
2194     { author fmt.names.all output.nonnull
2195       %new.sentence
2196     }
2197 ⟨/long⟩
2198 ⟨∗!long⟩
2199     {author num.names$ #4 <
2200       {
2201 ⟨strict⟩        author fmt.names.first output
2202 ⟨!strict⟩        author fmt.names.three output
2203        new.sentence
2204       }
2205      'skip$
2206    if$}
2207 ⟨/!long⟩
2208   if$
2209 }
2210
```

bookauthor.head   There are also 2 version of the function bookauthor.head. Not used anymore!

output.author.rest   Writes the rest of authors list after the entry title followed by a slash. In modern bibstyles, the list of authors in the heading part of a bibliographic record is cut to either 3 persons (if no !strict option is applied) or 1 person (if strict option). By idea, output.author.rest should print the rest of the authors in the author field but currently it also repeats the persons printed in the heading part. output.author.rest returns a null string ("") if all persons are printed in the heading part of the record. Note that all persons are alwats printed in the heading part if long option is applied.

```
2211 FUNCTION {output.author.rest}
2212 {
2213 ⟨∗!long⟩
2214   author empty$
2215     'skip$
2216 ⟨strict⟩     {author num.names$ #1 >
2217 ⟨!strict⟩     {author num.names$ #3 >
2218       { author fmt.names.all output
2219         new.semicolon
2220       }
2221      'skip$
2222    if$}
2223   if$
2224 ⟨/!long⟩
2225 }
2226
```

bookauthor.rest   This function is used only in `inbook` entry. It always cuts list to 4 persons since `format.bookauthors.rest` does that.

```
2227 FUNCTION {bookauthor.rest}
2228 {
2229   bookauthor empty$
2230     'skip$
2231     {
2232 ⟨strict⟩        bookauthor fmt.names.all output
2233 ⟨!strict⟩         bookauthor fmt.names.three output
2234       new.semicolon
2235     }
2236   if$
2237 }
2238
```

organization.rest

```
2239 FUNCTION {editor.organization.rest}
2240 {
2241   compiler empty$
2242     {}
2243     { format.compiler.rest   output
2244     new.semicolon
2245     }
2246   if$
2247   editor empty$
2248     {}
2249     { format.editors.rest   output.nonnull
2250     new.semicolon
2251     }
2252   if$
2253   organization empty$
2254     {}
2255     {organization output.nonnull
2256     new.semicolon
2257     }
2258   if$
2259 }
2260
```

format.url

```
2261 FUNCTION {format.url}
2262 { url empty$
2263     { "" }
2264     {
2265       bbl.url ": \BibUrl{" * url * "}" *
2266       urldate empty$
2267         { "" }
2268         { " (" bbl.urldate * ": " * urldate * ")" * }
2269       if$ *
2270     }
2271   if$
```

58

```
2272 }
2273
```

```
2274 FUNCTION {output.url}
2275 {
2276   url empty$
2277     'skip$
2278     { format.url output }
2279   if$
2280 }
2281
```

```
2282 FUNCTION {format.annote}
2283 { annote empty$
2284     { "" }
2285 { after.sentence 'output.state :=
2286   "\BibAnnote{" annote add.period$ * "}" *
2287 }
2288   if$
2289 }
2290
```

⟨NB: Do we really need to provide electronic search for ISBN?⟩

```
2291 FUNCTION {format.isbn}
2292 {
2293   isbn empty$
2294     { "" }
2295     { "ISBN:~\href{http://isbndb.com/search-all.html?kw=" isbn *
2296       "}{" * isbn * "}" *
2297     }
2298   if$
2299 }
2300
```

The Digital Object Identifier (DOI) System is for identifying content objects in the digital environment. DOI names are assigned to any entity for use on digital networks. They are used to provide current information, including where they (or information about them) can be found on the Internet. Information about a digital object may change over time, including where to find it, but its DOI name will not change.

Function `add.doi` embraces last string in stack into hyperlink that links it to specified `doi` identificator at https://doi.org/ web-site.

```
2301 ⟨∗eprint⟩
2302 FUNCTION {add.doi}
2303 { duplicate$ empty$
2304     'skip$
2305     { doi empty$
2306         'skip$
2307         { "\href{https://doi.org/" doi * "}{" * swap$ * "}" * }
2308       if$
```

```
2309       }
2310    if$
2311 }
2312 ⟨/eprint⟩
```

If `.bst` style is compiled without `eprint` option, we just ignore `doi` field.

```
2313 ⟨*!eprint⟩
2314 FUNCTION {add.doi} { }
2315 ⟨/!eprint⟩
2316
```

add.media   New in version 1.2. Adds `media` field if `strict` options is in effect. If the `media` field is empty `add.media` prints a value based on the `type` field. If the `type` is also empty `add.media` prints equivalent of the word `text` in current language.

```
2317 ⟨*!strict⟩
2318 FUNCTION {add.media}
2319 { duplicate$ empty$
2320     'skip$
2321     { media empty$
2322         'skip$
2323         { " " * bbl.media bracketise * }
2324       if$
2325     }
2326   if$
2327 }
2328 ⟨/!strict⟩
2329 ⟨*strict⟩
2330 FUNCTION {add.media}
2331 { duplicate$ empty$
2332     'skip$
2333     { media empty$
2334       { type$ "online" =
2335           { " " * bbl.media.online bracketise * }
2336           { " " * bbl.media.text   bracketise * }
2337         if$
2338       }
2339       { " " * bbl.media bracketise * }
2340     if$
2341     }
2342   if$
2343 }
2344 ⟨/strict⟩
2345
```

6.13   Electronic Publishing Information

The biblatex package provides three fields for electronic publishing information: `eprint`, `eprinttype`, and `eprintclass`. The `eprint` field is a verbatim field similar to `doi` which holds the identifier of the item. The `eprinttype` field holds the resource name, i. e., the name of the site or electronic archive. Optional `eprintclass` field is intended for additional information specific to the resource

indicated by the `eprinttype` field. This could be a section, a path, classification information, etc. If the `eprinttype` field is available, the standard styles will use it as a literal label. In the following example, they would print "Resource: identifier" rather than the generic "eprint: identifier":

```
eprint = {identifier},
eprinttype = {Resource},
```

format.eprint   The electronic identifier of an online publication. This is roughly comparable to a `doi` but specific to a certain archive, repository, service, or system. Also see eprinttype and eprintclass.
⟨NB: This function should use url. TO BE DONE YET.⟩

```
2346 ⟨∗eprint⟩
2347 %FUNCTION {format.eprint}
2348 %{ eprint empty$
2349 %   { "" }
2350 %   { eprintclass empty$
2351 %       { " \href{http://arxiv.org/abs/" eprint * "}" * "{" * eprint * "}" * }
2352 %       { eprinttype empty$
2353 %           { " \href{http://arxiv.org/abs/" eprint * "}" *
2354 %             "{" * eprintclass * "/" * eprint * "}" *
2355 %           }
2356 %           { " \href{http://arxiv.org/abs/" eprint * "}" *
2357 %             "{" * eprinttype * ":" * eprintclass * "/" * eprint * "}" *
2358 %           }
2359 %       if$}
2360 %   if$}
2361 %if$}
2362
2363 %FUNCTION {format.eprint}
2364 %{ eprint empty$
2365 %   { "" }
2366 %   { eprinttype empty$
2367 %       { "" }
2368 %       { eprinttype "~: " *}
2369 %     if$
2370 %     eprintclass empty$
2371 %       { }
2372 %       { eprintclass * "/" *}
2373 %     if$
2374 %     eprint *
2375 %   }
2376 %  if$
2377 %  url empty$
2378 %     { }
2379 %     { "\href{" url * "}{" * swap$ * "}" *}
2380 %  if$
2381 %}
2382
2383 FUNCTION {format.eprint}
2384 { eprint empty$
```

61

```
2385     { "" }
2386     { eprinttype empty$
2387         { "" }
2388         { eprinttype "~: " *}
2389       if$
2390       eprintclass empty$
2391         { }
2392         { eprintclass * "/" *}
2393       if$
2394       url empty$
2395         { eprint * }
2396         { "\href{" * url * "}{" * eprint * "}" *}
2397       if$
2398     }
2399   if$
2400 }
2401
2402 FUNCTION {output.eprint.url}
2403 {
2404   eprint empty$
2405     { format.url output }
2406     { format.eprint output }
2407   if$
2408 }
2409
2410 ⟨/eprint⟩
2411
2412 ⟨*!eprint⟩
2413 FUNCTION {output.eprint.url}
2414 {
2415     format.url output
2416 }
2417 ⟨/!eprint⟩
2418
```

Functions added in v1.2f to format patent entry (thanks to Stanislav Kruchinin).

add.number

```
2419 FUNCTION {add.number}
2420 { duplicate$ empty$
2421     { "" }
2422     { bbl.nr swap$ tie.or.space.connect }
2423   if$
2424 }
2425
```

ormat.type.number

```
2426 FUNCTION {format.type.number}
2427 {
2428   type empty$
2429     { "" }
2430     {
```

```
2431      number empty$
2432        { "" }
2433        { type number tie.or.space.connect }
2434     if$
2435     }
2436   if$
2437 }
2438
```

```
2439 FUNCTION {format.requestdate}
2440 { requestdate empty$
2441     { "" }
2442     { bbl.req requestdate tie.or.space.connect }
2443   if$
2444 }
2445
```

```
2446 FUNCTION {format.publicationdate}
2447 { publicationdate empty$
2448     { "" }
2449     { bbl.publ publicationdate tie.or.space.connect }
2450   if$
2451 }
2452
```

```
2453 FUNCTION {format.prioritydate}
2454 { prioritydate empty$
2455     { "" }
2456     { bbl.priority prioritydate tie.or.space.connect }
2457   if$
2458 }
2459
```

6.14 Entry types

Text below in this section is borrowed from biblatex manual. Not every field listed below is actually supported by GOST styles. So description below should be considered as a goal or a feature request.

The lists below indicate the fields supported by each entry type. Note that the mapping of fields to an entry type is ultimately at the discretion of the bibliography style. The lists below therefore serve two purposes. They indicate the fields supported by the standard styles which ship with this package and they also serve as a model for custom styles. Note that the required fields are not strictly required in all cases. The fields marked as optional are optional in a technical sense. Bibliographical formatting rules usually require more than just the required fields. The standard styles will generally not perform any formal validity checks, but custom styles may do so. Generic fields like abstract and annotation or label and shorthand are not included in the lists below because they are independent of the entry type.

6.14.1 Regular Types

article   An article in a journal, magazine, newspaper, or other periodical which forms a self-contained unit with its own title. The title of the periodical is given in the journaltitle field. If the issue has its own title in addition to the main title of the periodical, it goes in the issuetitle field. Note that editor and related fields refer to the journal while translator and related fields refer to the article.

    Required fields: author, title, journaltitle, year/date.

    Optional fields: translator, annotator, commentator, subtitle, titleaddon, editor, editora, editorb, editorc, journalsubtitle, issuetitle, issuesubtitle, language, origlanguage, series, volume, number, eid, issue, month, pages, version, note, issn, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2460 FUNCTION {article}
2461 {
2462   output.bibitem
2463   output.author.head
2464   %new.sentence
2465 ⟨natbib⟩  author format.key output
2466   title add.media "title" output.check
2467   new.slash
2468   output.author.rest
2469   new.dblslash
2470   journal emphasize add.doi "journal" output.check % new in v1.2
2471   new.block
2472   format.date "year/date" output.check
2473   new.block
2474   format.volume output
2475   format.number output
2476   new.block
2477   format.pages.page output
2478   new.block
2479   note output
2480   new.sentence
2481 %  format.url output
2482   output.eprint.url
2483   format.annote output
2484   fin.entry
2485 }
2486
```

book   A single-volume book with one or more authors where the authors share credit for the work as a whole. In biblatex, this entry type also covers the function of the @inbook type of traditional BibTeX.

    Required fields: author, title, year/date.

    Optional fields: editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2487 FUNCTION {book}
2488 {
```

```
2489    output.bibitem
2490    output.author.head
2491    new.sentence
2492 ⟨natbib⟩   author format.key output
2493    title add.doi add.media "title" output.check
2494    new.colon            % added in v.1.2k
2495    titleaddon output  % added in v.1.2k
2496    new.slash
2497    output.author.rest
2498    editor.organization.rest
2499    new.sentence
2500    format.number.series output
2501    new.block
2502    format.edition output
2503    new.block
2504    output.address.publisher
2505    format.date "year/date" output.check
2506    new.block
2507    format.bvolume output
2508    new.block
2509    format.pages.page output
2510    new.block
2511 ⟨eprint⟩   format.isbn output
2512 ⟨eprint⟩   new.block
2513    note output
2514    new.sentence
2515 %   format.url output
2516    output.eprint.url
2517    format.annote output
2518    fin.entry
2519 }
2520
```

booklet   A book-like work without a formal publisher or sponsoring institution. Use the field
              howpublished to supply publishing information in free format, if applicable. The field type may
              be useful as well.
                    Required fields: author/editor, title, year/date.
                    Optional fields: subtitle, titleaddon, language, howpublished, type, note, location, chapter, pages,
              pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2521 FUNCTION {booklet}
2522 {
2523    output.bibitem
2524    output.author.head
2525    new.sentence
2526 ⟨natbib⟩   author format.key output
2527    title add.doi add.media "title" output.check
2528    new.colon            % added in v.1.2k
2529    titleaddon output  % added in v.1.2k
2530    new.slash
2531    output.author.rest
```

```
2532   editor.organization.rest
2533   new.block
2534   howpublished output
2535   address.or.location output
2536   format.date "year/date" output.check
2537   new.block
2538   note output
2539   new.sentence
2540 %  format.url output
2541   output.eprint.url
2542   format.annote output
2543   fin.entry
2544 }
2545
```

inbook   A part of a book which forms a self-contained unit with its own title. Note that the profile of this entry type is different from standard BibTeX.

     Required fields: author, title, booktitle, year/date.

     Optional fields: bookauthor, editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2546 FUNCTION {inbook}
2547 {
2548   output.bibitem
2549   output.author.head
2550   new.sentence
2551 ⟨natbib⟩   author format.key output
2552   title add.doi add.media "title" output.check
2553   new.colon          % added in v.1.2k
2554   titleaddon output  % added in v.1.2k
2555   new.slash
2556   output.author.rest
2557   new.dblslash
2558 %  bookauthor.head
2559   booktitle "booktitle" output.check
2560   new.slash
2561   bookauthor.rest
2562   editor.organization.rest
2563   new.block
2564   format.edition output
2565   new.block
2566   format.number.series output
2567   new.sentence
2568   output.address.publisher
2569   format.date "year/date" output.check
2570   new.block
2571   format.bvolume output
2572   new.block
```

```
2573   format.pages.page output
2574   new.block
2575 ⟨eprint⟩   format.isbn output
2576 ⟨eprint⟩   new.block
2577   note output
2578   new.sentence
2579 %  format.url output
2580   output.eprint.url
2581   format.annote output
2582   fin.entry
2583 }
2584
```

incollection A contribution to a collection which forms a self-contained unit with a distinct author and title. The author refers to the title, the editor to the booktitle, i. e., the title of the collection.

  Required fields: author, editor, title, booktitle, year/date.

  Optional fields: editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2585 FUNCTION {incollection}
2586 {
2587   output.bibitem
2588   output.author.head
2589   new.sentence
2590 ⟨natbib⟩   author format.key output
2591   new.sentence
2592   title add.doi add.media "title" output.check
2593   new.colon          % added in v.1.2k
2594   titleaddon output  % added in v.1.2k
2595   new.slash
2596   output.author.rest
2597   new.dblslash
2598   booktitle "booktitle" output.check
2599   new.slash
2600   editor.organization.rest
2601   new.block
2602   output.address.publisher
2603   format.date "year/date" output.check
2604   new.block
2605   format.bvolume output
2606   format.number.series output
2607   new.block
2608   format.pages.page output
2609   new.block
2610   note output
2611   new.sentence
2612 %  format.url output
2613   output.eprint.url
```

```
2614    format.annote output
2615    fin.entry
2616 }
2617
```

proceedings   A single-volume conference proceedings. This type is very similar to @collection. It supports an optional organization field which holds the sponsoring institution. The editor is omissible.

Required fields: editor, title, year/date.

Optional fields: subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, eventtitle, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2618 FUNCTION {proceedings}
2619 {
2620    output.bibitem
2621 ⟨natbib⟩   editor format.key output
2622    title add.doi add.media "title" output.check
2623    new.colon                    % added in v.1.2k
2624    titleaddon output            % added in v.1.2k
2625    new.slash
2626    editor.organization.rest
2627    new.block
2628    output.address.publisher     % 1.2k moved from below
2629    new.block                    % added in v.1.2k
2630    format.date "year/date" output.check
2631    new.block
2632    format.bvolume output
2633    format.number.series output
2634    new.block
2635    format.pages.page output
2636 %%%output.address.publisher   % 1.2k moved upper
2637    new.block
2638    note output
2639    new.sentence
2640 %  format.url output
2641    output.eprint.url
2642    format.annote output
2643    fin.entry
2644 }
2645
```

inproceedings   An article in a conference proceedings. This type is similar to @incollection. It supports an optional organization field.

Required fields: author, editor, title, booktitle, year/date.

Optional fields: subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, eventtitle, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2646 FUNCTION {inproceedings}
2647 { output.bibitem
```

```
2648    output.author.head
2649    new.sentence
2650 ⟨natbib⟩  author format.key output
2651    new.sentence
2652    title add.doi add.media "title" output.check
2653    new.colon                % added in v.1.2k
2654    titleaddon output        % added in v.1.2k
2655    new.slash
2656    output.author.rest
2657    new.dblslash
2658    booktitle "booktitle" output.check
2659    new.slash
2660    editor.organization.rest
2661    new.block
2662    output.address.publisher   % 1.2k moved from below
2663    new.block                  % added in v.1.2k
2664 %  address empty$
2665 %    { publisher output
2666 %      format.date "year/date" output.check
2667 %    }
2668 %    { address output.nonnull
2669 %      new.colon
2670 %      publisher output
2671 %      format.date "year/date" output.check
2672 %    }
2673 %  if$
2674 %  output.address.publisher        % moved upper in v.1.2k
2675    format.date "year/date" output.check
2676    new.block
2677    format.bvolume output
2678    format.number.series output
2679    new.block
2680    format.pages.page output
2681    new.block
2682    note output
2683    new.sentence
2684 %  format.url output
2685    output.eprint.url
2686    format.annote output
2687    fin.entry
2688 }
2689
```

manual   Technical or other documentation, not necessarily in printed form. The author or editor is omissible.
          Required fields: author/editor, title, year/date.
          Optional fields: subtitle, titleaddon, language, edition, type, series, number, version, note,
      organization, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint,
      eprintclass, eprinttype, url, urldate.

```
2690 FUNCTION {manual}
2691 { output.bibitem
```

```
2692    author empty$
2693      { organization empty$
2694          'skip$
2695          { organization output.nonnull
2696            address output
2697          }
2698        if$
2699      }
2700      { format.author output.nonnull }
2701    if$
2702 ⟨natbib⟩  author format.key output
2703    new.block
2704    title add.doi add.media "title" output.check
2705    new.colon            % added in v.1.2k
2706    titleaddon output  % added in v.1.2k
2707    author empty$
2708      { organization empty$
2709      {
2710            address new.block.checka
2711            address output
2712        }
2713          'skip$
2714        if$
2715      }
2716      {
2717        organization address new.block.checkb
2718        organization output
2719        address output
2720      }
2721    if$
2722    format.edition output
2723    format.date "year/date" output.check
2724    new.block
2725    note output
2726    new.sentence
2727 %  format.url output
2728    output.eprint.url
2729    format.annote output
2730    fin.entry
2731 }
2732
```

patent   A patent or patent request. The number or record token is given in the number field. Use the `type` field to specify the type and the `location` field to indicate the scope of the patent, if different from the scope implied by the type. Note that the location field is treated as a key list with this entry type.

Required fields: author, title, number, year/date.

Optional fields: holder, subtitle, titleaddon, type, version, location, note, date, month, year, addendum, pubstate, doi, eprint, eprint class, eprint type, url, urldate.

```
2733 FUNCTION {patent}
```

```
2734 {
2735   output.bibitem
2736   title add.media output.nonnull
2737   new.colon            % added in v.1.2k
2738   titleaddon output  % added in v.1.2k
2739   new.colon
2740   format.type.number output
2741   add.blank
2742   address.or.location output
2743   new.colon
2744   ipc output
2745   new.slash
2746   format.author "author" output.check
2747   add.blank
2748   authorcountry paranthesify output.nonnull
2749 ⟨natbib⟩   author format.key output
2750   new.semicolon
2751   holder output.nonnull
2752   new.semicolon
2753   credits output.nonnull
2754   new.block
2755   requestnumber add.number output
2756   new.semicolon
2757   format.requestdate output
2758   new.semicolon
2759   format.publicationdate output
2760   publication output
2761   new.semicolon
2762   format.prioritydate output
2763   prioritynumber output
2764   prioritycountry paranthesify output
2765   new.block
2766   note output
2767   new.sentence
2768   output.url
2769   format.annote output
2770   new.block
2771   pagetotal output
2772   fin.entry
2773 }
2774
```

misc   A fallback type for entries which do not fit into any other category. Use the field howpublished to supply publishing information in free format, if applicable. The field type may be useful as well. author, editor, and year are omissible.

   Required fields: author/editor, title, year/date.

```
2775 FUNCTION {misc}
2776 { output.bibitem
2777   %format.author output  % < v.1.2k
2778   output.author.head     % v.1.2k
```

71

```
2779    %new.sentence              % v.1.2k
2780 ⟨natbib⟩  author format.key output
2781    title howpublished new.sentence.checkb
2782    title add.media output
2783    howpublished new.block.checka
2784    howpublished output
2785    new.block
2786    format.date "year/date" output.check
2787    new.block
2788    note output
2789    new.sentence
2790 %  format.url output
2791    output.eprint.url
2792    format.annote output
2793    fin.entry
2794 }
2795
```

unpublished   A work with an author and a title which has not been formally published, such as a manuscript or the script of a talk. Use the fields howpublished and note to supply additional information in free format, if applicable.

Required fields: author, title, year/date.

Optional fields: subtitle, titleaddon, language, howpublished, note, location, isbn, date, month, year, addendum, pubstate, url, urldate

```
2796 FUNCTION {unpublished}
2797 { output.bibitem
2798   output.author.head
2799   new.sentence
2800 ⟨natbib⟩  author format.key output
2801   title "title" output.check
2802   new.colon          % added in v.1.2k
2803   titleaddon output  % added in v.1.2k
2804   new.slash
2805   output.author.rest
2806   editor.organization.rest
2807   new.block
2808   format.date "year/date" output.check
2809   new.block
2810   note "note" output.check
2811   new.sentence
2812 %  format.url output
2813   output.eprint.url
2814   format.annote output
2815   fin.entry
2816 }
2817
```

online   An online resource. Author, editor, and year are omissible. This entry type is intended for sources such as web sites which are intrinsically online resources. Note that all entry types support the url field. For example, when adding an article from an online journal, it may be preferable to use the @article type and its url field.

Required fields: author/editor, title, year/date, url.

Optional fields: subtitle, titleaddon, language, version, note, organization, date, month, year, addendum, pubstate, urldate.

```
2818 FUNCTION {online}
2819 { output.bibitem
2820   format.author output
2821 ⟨natbib⟩   author format.key output
2822   title howpublished new.sentence.checkb
2823   title add.doi add.media "title" output.check
2824   new.colon           % added in v.1.2k
2825   titleaddon output   % added in v.1.2k
2826 %  howpublished new.block.checka
2827   howpublished new.dblslash.checka
2828 ⟨!strict⟩   howpublished output
2829 ⟨strict⟩   howpublished bracketise output
2830   editor.organization.rest
2831   new.sentence
2832   new.block
2833   output.address.publisher
2834   format.date output
2835   new.block
2836 %  format.url output
2837   output.eprint.url
2838   new.sentence
2839   note output
2840   format.annote output
2841   fin.entry
2842 }
2843
```

`internet`   New in version 2012.02.15.

```
   www 2844 FUNCTION {internet}   {online}
webpage 2845 FUNCTION {www}        {online}
ielectronic 2846 FUNCTION {webpage}    {online}
        2847 FUNCTION {electronic} {online}
```

`thesis`   New in version 2012.02.02.

A thesis written for an educational institution to satisfy the requirements for a degree. Use the type field to specify the type of thesis.

Required fields: author, title, type, institution, year/date.

Optional fields: subtitle, titleaddon, language, note, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

```
2848 FUNCTION {thesis}
2849 { output.bibitem
2850   format.author "author" output.check
2851 ⟨natbib⟩   author format.key output
2852   new.sentence
2853   title add.doi add.media "title" output.check
2854   new.colon
2855   bbl.phdthesis format.thesis.type output.nonnull
```

73

```
2856   new.colon
2857   specialitycode.or.number output % code of the speciality, new in v.1.2i
2858   new.colon
2859   titleaddon output % date of defence and approvement; new in v.1.2i
2860   new.slash
2861   %%%format.author.rest output %% dupplicates athours from the head zone
2862   output.author.rest              %% prints if num.names$ > 3 or > 1
2863   new.semicolon
2864   %institution "institution" output.check
2865   institution.or.school "institution/school" output.check
2866   new.block
2867   output.address.publisher
2868   format.date "year/date" output.check
2869   new.block
2870   format.pages.page output
2871   new.block
2872   note output
2873   new.sentence
2874 %  format.url output
2875   output.eprint.url
2876   format.annote output
2877   fin.entry
2878 }
```

report   New in version 2012.02.02.

A technical report, research report, or white paper published by a university or some other institution. Use the type field to specify the type of report. The sponsoring institution goes in the institution field.

Required fields: author, title, type, institution, year/date.

Optional fields: subtitle, titleaddon, language, number, version, note, location, month, isrn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate.

```
2879 %FUNCTION {report}
2880 %{
2881 %  output.bibitem
2882 %  output.author.head
2883 %  new.sentence
2884 %  title add.doi add.media "title" output.check
2885 %  new.colon
2886 %%  format.techrep.type.number output.nonnull
2887 %  type "type" output.check
2888 %  new.slash
2889 %  output.author.rest
2890 %  editor.organization.rest
2891 %  new.block
2892 %  address output
2893 %  new.colon
2894 %  institution "institution" output.check
2895 %  format.date "year/date" output.check
2896 %  new.block
2897 %  note output
```

```
2898 %   new.block                    % v.2
2899 %   format.pages.page output % v.2
2900 %   new.sentence
2901 %%  format.url output
2902 %   output.eprint.url
2903 %   format.annote output
2904 %   fin.entry
2905 %}
2906 FUNCTION {report}
2907 {
2908   output.bibitem
2909 %   output.author.head
2910 %   new.sentence
2911   title add.doi add.media "title" output.check
2912 ⟨natbib⟩   title format.key output
2913   new.colon           % added in v.1.2k
2914   titleaddon output  % added in v.1.2k
2915   new.colon
2916 %  format.techrep.type.number output.nonnull
2917 %  type "type" output.check
2918 %  format.report.type.number "type" output.check
2919   type "type" output.check
2920   new.colon
2921   number output
2922   new.slash
2923   %institution "institution" output.check
2924   institution.or.school "institution/school" output.check
2925   new.semicolon
2926   format.chief.rest output % from editor field
2927   new.semicolon
2928   format.executor.rest output % from author field
2929   new.block
2930   address.or.location output
2931   new.colon
2932   organization output
2933   format.date "year/date" output.check
2934   new.block                 % v.2
2935   format.pages.page output % v.2
2936   new.sentence % или new.block ?
2937   output.eprint.url
2938   new.block
2939   note output
2940   format.annote output
2941   fin.entry
2942 }
2943
```

6.14.2 Type Aliases

The entry types listed in this section are provided for backwards compatibility with traditional BibTeX styles. These aliases are resolved by BibTeX as the data is exported. Bibliography styles will see the entry type the alias points to, not the alias name. All unknown entry types are generally exported as @misc.

phdthesis   Similar to @thesis except that the type field is optional and defaults to the localized term 'PhD thesis'. You may still use the type field to override that.

```
2944 FUNCTION {phdthesis}
2945 { output.bibitem
2946   format.author "author" output.check
2947 ⟨natbib⟩   author format.key output
2948   new.sentence
2949   title add.doi add.media "title" output.check
2950   new.colon
2951   bbl.phdthesis format.thesis.type output.nonnull
2952   new.colon
2953   %%number output % code of the speciality
2954   specialitycode.or.number output % code of the speciality, new in v.1.2i
2955   new.colon
2956   titleaddon output % date of defence and approvement; new in v.1.2i
2957   new.slash
2958   %%%format.author.rest output %% dupplicates athours
2959   output.author.rest             %% prints if num.names$ > 3 or > 1
2960   new.semicolon
2961   %institution "institution" output.check
2962   institution.or.school "institution/school" output.check
2963   new.block
2964   output.address.publisher
2965   format.date "year/date" output.check
2966   new.block
2967   format.pages.page output
2968   new.block
2969   note output
2970   new.sentence
2971 %  format.url output
2972   output.eprint.url
2973   format.annote output
2974   fin.entry
2975 }
```

mastersthesis   Similar to @thesis except that the type field is optional and defaults to the localized term 'Master's thesis'. You may still use the type field to override that.

```
2976 FUNCTION {mastersthesis}
2977 { output.bibitem
2978   format.author "author" output.check
2979 ⟨natbib⟩   author format.key output
2980   new.sentence
2981   title add.doi add.media "title" output.check
2982   new.colon
```

```
2983    bbl.mathesis format.thesis.type output.nonnull
2984    new.colon
2985    %%number output % code of the speciality
2986    specialitycode.or.number output % code of the speciality, new in v.1.2i
2987    new.colon
2988    titleaddon output % date of defence and approvement; new in v.1.2i
2989    new.slash
2990    %%%format.author.rest output %% dupplicates athours
2991    output.author.rest              %% prints if num.names$ > 3 or > 1
2992    new.semicolon
2993    %institution "institution" output.check
2994    institution.or.school "institution/school" output.check
2995    new.block
2996    output.address.publisher
2997    format.date "year/date" output.check
2998    new.block
2999    format.pages.page output
3000    new.block
3001    note output
3002    new.sentence
3003 %  format.url output
3004    output.eprint.url
3005    format.annote output
3006    fin.entry
3007 }
```

docthesis   Similar to @thesis except that the type field is optional and defaults to the localized term 'Doctor's of sciences thesis'. You may still use the type field to override that.

```
3008 FUNCTION {docthesis}
3009 { output.bibitem
3010    format.author "author" output.check
3011 ⟨natbib⟩  author format.key output
3012    new.sentence
3013    title add.doi add.media "title" output.check
3014    new.colon
3015    bbl.docthesis format.thesis.type output.nonnull
3016    new.colon
3017    %%number output % code of the speciality
3018    specialitycode.or.number output % code of the speciality, new in v.1.2i
3019    new.colon
3020    titleaddon output % date of defence and approvement; new in v.1.2i
3021    new.slash
3022    %%%format.author.rest output %% dupplicate athours
3023    output.author.rest              %% prints if num.names$ > 3 or > 1
3024    new.semicolon
3025    %institution "institution" output.check
3026    institution.or.school "institution/school" output.check
3027    new.block
3028    output.address.publisher
3029    format.date "year/date" output.check
3030    new.block
```

77

```
3031    format.pages.page output
3032    new.block
3033    note output
3034    new.sentence
3035 %  format.url output
3036    output.eprint.url
3037    format.annote output
3038    fin.entry
3039 }
```

conference

```
3040 FUNCTION {conference} { inproceedings }
3041
```

techreport   TechReport is similar to @report except that the type field is optional and defaults to the localized term 'technical report'. You may still use the type field to override that.

```
3042 %FUNCTION {techreport}
3043 %{
3044 %  output.bibitem
3045 %  output.author.head
3046 %  new.sentence
3047 %  title add.doi add.media "title" output.check
3048 %  new.colon
3049 %  format.techrep.type.number output.nonnull
3050 %  new.slash
3051 %  output.author.rest
3052 %  editor.organization.rest
3053 %  new.block
3054 %  address output
3055 %  new.colon
3056 %  institution "institution" output.check
3057 %  format.date "year/date" output.check
3058 %  new.block
3059 %  note output
3060 %  new.block                   % v.2
3061 %  format.pages.page output % v.2
3062 %  new.sentence
3063 %%  format.url output
3064 %  output.eprint.url
3065 %  format.annote output
3066 %  fin.entry
3067 %}
3068
3069 FUNCTION {techreport}
3070 {
3071    output.bibitem
3072 %  output.author.head
3073 %  new.sentence
3074    title add.doi add.media "title" output.check
3075 ⟨natbib⟩   title format.key output
3076    new.colon
```

```
3077 %  format.techrep.type.number output.nonnull
3078 %  type "type" output.check
3079 %  format.report.type.number "type" output.check
3080 %  type output
3081    format.techreport.type output
3082    new.colon
3083    number output
3084    new.slash
3085    %institution "institution" output.check
3086    institution.or.school "institution/school" output.check
3087    new.semicolon
3088    format.chief.rest output % from editor field
3089    new.semicolon
3090    format.executor.rest output % from author field
3091    new.block
3092    address.or.location output
3093    new.colon
3094    organization output
3095    format.date "year/date" output.check
3096    new.block            % v.2
3097    format.pages.page output % v.2
3098    new.sentence % или new.block ?
3099    output.eprint.url
3100    new.block
3101    note output
3102    format.annote output
3103    fin.entry
3104 }
3105
```

default.type

```
3106 FUNCTION {default.type} { misc }
3107
```

6.15 Month Abbreviations

Borrowed from `merlin.mbs` of package custom-bib. This is done for backward compatibility with standard .bst styles which are designed for English. The string in the definition of any month macro must coincide with that used in `format.month` function in the above.

```
3108 MACRO {jan} {"Jan."}
3109 MACRO {feb} {"Feb."}
3110 MACRO {mar} {"Mar."}
3111 MACRO {apr} {"Apr."}
3112 MACRO {may} {"May"}
3113 MACRO {jun} {"Jun."}
3114 MACRO {jul} {"Jul."}
3115 MACRO {aug} {"Aug."}
3116 MACRO {sep} {"Sep."}
3117 MACRO {oct} {"Oct."}
3118 MACRO {nov} {"Nov."}
```

3119 `MACRO {dec} {"Dec."}`

6.16 Journal Abbreviations

6.16.1 Physics and astronomy

Borrowed from `physjour.mbs` of package custom-bib.

```
3120 MACRO {aa}{"Astron.\ \& Astrophys."}
3121 MACRO {aasup}{"Astron.\ \& Astrophys.\ Suppl.\ Ser."}
3122 MACRO {aj} {"Astron.\ J."}
3123 MACRO {aph} {"Acta Phys."}
3124 MACRO {advp} {"Adv.\ Phys."}
3125 MACRO {ajp} {"Amer.\ J.\ Phys."}
3126 MACRO {ajm} {"Amer.\ J.\ Math."}
3127 MACRO {amsci} {"Amer.\ Sci."}
3128 MACRO {anofd} {"Ann.\ Fluid Dyn."}
3129 MACRO {am} {"Ann.\ Math."}
3130 MACRO {ap} {"Ann.\ Phys.\ (NY)"}
3131 MACRO {adp} {"Ann.\ Phys.\ (Leipzig)"}
3132 MACRO {ao} {"Appl.\ Opt."}
3133 MACRO {apl} {"Appl.\ Phys.\ Lett."}
3134 MACRO {app} {"Astroparticle Phys."}
3135 MACRO {apj} {"Astrophys.\ J."}
3136 MACRO {apjsup} {"Astrophys.\ J.\ Suppl."}
3137 MACRO {apss} {"Astrophys.\ Space Sci."}
3138 MACRO {araa} {"Ann.\ Rev.\ Astron.\ Astrophys."}
3139 MACRO {baas} {"Bull.\ Amer.\ Astron.\ Soc."}
3140 MACRO {baps} {"Bull.\ Amer.\ Phys.\ Soc."}
3141 MACRO {cmp} {"Comm.\ Math.\ Phys."}
3142 MACRO {cpam} {"Commun.\ Pure Appl.\ Math."}
3143 MACRO {cppcf} {"Comm.\ Plasma Phys.\ \& Controlled Fusion"}
3144 MACRO {cpc} {"Comp.\ Phys.\ Comm."}
3145 MACRO {cqg} {"Class.\ Quant.\ Grav."}
3146 MACRO {cra} {"C.\ R.\ Acad.\ Sci.\ A"}
3147 MACRO {fed} {"Fusion Eng.\ \& Design"}
3148 MACRO {ft} {"Fusion Tech."}
3149 MACRO {grg} {"Gen.\ Relativ.\ Gravit."}
3150 MACRO {ieeens} {"IEEE Trans.\ Nucl.\ Sci."}
3151 MACRO {ieeeps} {"IEEE Trans.\ Plasma Sci."}
3152 MACRO {ijimw} {"Interntl.\ J.\ Infrared \& Millimeter Waves"}
3153 MACRO {ip} {"Infrared Phys."}
3154 MACRO {irp} {"Infrared Phys."}
3155 MACRO {jap} {"J.\ Appl.\ Phys."}
3156 MACRO {jasa} {"J.\ Acoust.\ Soc.\ America"}
3157 MACRO {jcp} {"J.\ Comp.\ Phys."}
3158 MACRO {jchp} {"J.\ Chem.\ Phys."}
3159 MACRO {jetp} {"Sov.\ Phys.--JETP"}
3160 MACRO {jfe} {"J.\ Fusion Energy"}
3161 MACRO {jfm} {"J.\ Fluid Mech."}
3162 MACRO {jmp} {"J.\ Math.\ Phys."}
```

```
3163 MACRO {jne} {"J.\ Nucl.\ Energy"}
3164 MACRO {jnec} {"J.\ Nucl.\ Energy, C: Plasma Phys., Accelerators, Thermonucl.\ Res."}
3165 MACRO {jnm} {"J.\ Nucl.\ Mat."}
3166 MACRO {jpc} {"J.\ Phys.\ Chem."}
3167 MACRO {jpp} {"J.\ Plasma Phys."}
3168 MACRO {jpsj} {"J.\ Phys.\ Soc.\ Japan"}
3169 MACRO {jsi} {"J.\ Sci.\ Instrum."}
3170 MACRO {jvst} {"J.\ Vac.\ Sci.\ \& Tech."}
3171 MACRO {nat} {"Nature"}
3172 MACRO {nature} {"Nature"}
3173 MACRO {nedf} {"Nucl.\ Eng.\ \& Design/Fusion"}
3174 MACRO {nf} {"Nucl.\ Fusion"}
3175 MACRO {nim} {"Nucl.\ Inst.\ \& Meth."}
3176 MACRO {nimpr} {"Nucl.\ Inst.\ \& Meth.\ in Phys.\ Res."}
3177 MACRO {np} {"Nucl.\ Phys."}
3178 MACRO {npb} {"Nucl.\ Phys.\ B"}
3179 MACRO {nt/f} {"Nucl.\ Tech./Fusion"}
3180 MACRO {npbpc} {"Nucl.\ Phys.\ B (Proc.\ Suppl.)"}
3181 MACRO {inc} {"Nuovo Cimento"}
3182 MACRO {nc} {"Nuovo Cimento"}
3183 MACRO {pf} {"Phys.\ Fluids"}
3184 MACRO {pfa} {"Phys.\ Fluids A: Fluid Dyn."}
3185 MACRO {pfb} {"Phys.\ Fluids B: Plasma Phys."}
3186 MACRO {pl} {"Phys.\ Lett."}
3187 MACRO {pla} {"Phys.\ Lett.\ A"}
3188 MACRO {plb} {"Phys.\ Lett.\ B"}
3189 MACRO {prep} {"Phys.\ Rep."}
3190 MACRO {pnas} {"Proc.\ Nat.\ Acad.\ Sci.\ USA"}
3191 MACRO {pp} {"Phys.\ Plasmas"}
3192 MACRO {pop} {"Phys.\ Plasmas"}
3193 MACRO {ppcf} {"Plasma Phys.\ \& Controlled Fusion"}
3194 MACRO {phitrsl} {"Philos.\ Trans.\ Roy.\ Soc.\ London"}
3195 MACRO {prl} {"Phys.\ Rev.\ Lett."}
3196 MACRO {pr} {"Phys.\ Rev."}
3197 MACRO {physrev} {"Phys.\ Rev."}
3198 MACRO {pra} {"Phys.\ Rev.\ A"}
3199 MACRO {prb} {"Phys.\ Rev.\ B"}
3200 MACRO {prc} {"Phys.\ Rev.\ C"}
3201 MACRO {prd} {"Phys.\ Rev.\ D"}
3202 MACRO {pre} {"Phys.\ Rev.\ E"}
3203 MACRO {ps} {"Phys.\ Scripta"}
3204 MACRO {procrsl} {"Proc.\ Roy.\ Soc.\ London"}
3205 MACRO {rmp} {"Rev.\ Mod.\ Phys."}
3206 MACRO {rsi} {"Rev.\ Sci.\ Inst."}
3207 MACRO {science} {"Science"}
3208 MACRO {sciam} {"Sci.\ Am."}
3209 MACRO {sam} {"Stud.\ Appl.\ Math."}
3210 MACRO {st} {"Sky and Telesc."}
```

6.16.2 Supplementary Journal Names

Borrowed from `suppjour.mbs` of package `custom-bib`.

```
3211 MACRO {cjp} {"Czech. J. Phys."}
3212 MACRO {el} {"Europhys. Lett."}
3213 MACRO {en} {"Europhys. News"}
3214 MACRO {fujitsustj} {"FUJITSU Sci. Tech. J."}
3215 MACRO {ieeeed} {"IEEE Trans. Electron Devices"}
3216 MACRO {ieeeim} {"IEEE Trans. Instrum. Meas."}
3217 MACRO {ieeejqe} {"IEEE J. Quantum Electron."}
3218 MACRO {ieeem} {"IEEE Trans. Magn."}
3219 MACRO {ieeeptl} {"IEEE Photonic Technol. Lett."}
3220 MACRO {ieeeuffc} {"IEEE Trans. Ultrason., Ferroelect., Freq. Cont."}
3221 MACRO {jem} {"J. Electron. Mater."}
3222 MACRO {jes} {"J. Electrochem. Soc."}
3223 MACRO {jetplett} {"JETP Lett."}
3224 MACRO {jjap} {"Japan. J. Appl. Phys."}
3225 MACRO {jpha} {"J. Phys. A: Math. Gen."}
3226 MACRO {jphb} {"J. Phys. B: At. Mol. Opt. Phys."}
3227 MACRO {jphbold} {"J. Phys. B: At. Mol. Phys."}
3228 MACRO {jphc} {"J. Phys.: Condens. Matter"}
3229 MACRO {jphcold} {"J. Phys. C: Solid State Phys."}
3230 MACRO {jphd} {"J. Phys. D: Appl. Phys."}
3231 MACRO {jvsta} {"J. Vac. Sci. Technol. A"}
3232 MACRO {jvstb} {"J. Vac. Sci. Technol. B"}
3233 MACRO {me} {"Microelectron. Eng."}
3234 MACRO {necrd} {"NEC Res.{\&} Develop."}
3235 MACRO {pa} {"Physica A"}
3236 MACRO {pb} {"Physica B"}
3237 MACRO {pc} {"Physica C"}
3238 MACRO {pd} {"Physica D"}
3239 MACRO {procieee} {"Proc. IEEE"}
3240 MACRO {procspie} {"Proc. SPIE"}
3241 MACRO {pssa} {"Phys. Stat. Sol. A"}
3242 MACRO {pssb} {"Phys. Stat. Sol. B"}
3243 MACRO {rpp} {"Rep. Progr. Phys."}
3244 MACRO {sm} {"Synthet. Metal"}
3245 MACRO {sost} {"Solid State Technol."}
3246 MACRO {ss} {"Surf. Sci."}
3247 MACRO {ssc} {"Solid State Commun."}
3248 MACRO {sst} {"Semicond. Sci. Technol."}
3249 MACRO {suplatt} {"Superlatt. Microstr."}
3250 MACRO {sust} {"Supercond. Sci. Technol."}
3251 MACRO {znat} {"Z. Naturforsch."}
```

6.16.3 Optics

Borrowed from `photjour.mbs`.

```
3252 MACRO {appopt} {"Appl.\ Opt."}
3253 MACRO {bell}   {"Bell Syst.\ Tech.\ J."}
```

```
3254 MACRO {ell}    {"Electron.\ Lett."}
3255 MACRO {jasp}   {"J.\ Appl.\ Spectr."}
3256 MACRO {jqe}    {"IEEE J.\ Quantum Electron."}
3257 MACRO {jlwt}   {"J.\ Lightwave Technol."}
3258 MACRO {jmo}    {"J.\ Mod.\ Opt."}
3259 MACRO {josa}   {"J.\ Opt.\ Soc.\ America"}
3260 MACRO {josaa}  {"J.\ Opt.\ Soc.\ Amer.~A"}
3261 MACRO {josab}  {"J.\ Opt.\ Soc.\ Amer.~B"}
3262 MACRO {jdp}    {"J.\ Phys.\ (Paris)"}
3263 MACRO {oc}     {"Opt.\ Commun."}
3264 MACRO {ol}     {"Opt.\ Lett."}
3265 MACRO {os}     {"Opt.\ Spectrosc."}
3266 MACRO {phtl}   {"IEEE Photon. Technol. Lett."}
3267 MACRO {pspie}  {"Proc.\ Soc.\ Photo-Opt.\ Instrum. Eng."}
3268 MACRO {vr}     {"Vision Res."}
3269 MACRO {zph}    {"Z.\ f.\ Physik"}
3270 MACRO {zphb}   {"Z.\ f.\ Physik~B"}
3271 MACRO {zphd}   {"Z.\ f.\ Physik~D"}
```

6.16.4   Physics of condensed Matter

```
3272 MACRO {sse} {"Solid-State Electron."}
3273 MACRO {pss} {"Phys. Sol. State"}
3274 MACRO {nl}  {"Nano Lett."}
```

6.16.5   Soviet and Russian journals

⟨NB: To be extended.⟩

```
3275 MACRO {sjpp} {"Sov.\ J.\ Plasma Phys."}
3276 MACRO {spd}  {"Sov.\ Phys.--Doklady"}
3277 MACRO {sptp} {"Sov.\ Phys.--Tech. Phys."}
3278 MACRO {spu}  {"Sov.\ Phys.--Uspekhi"}
3279 ⟨!utf8⟩MACRO {ufn}   {"\CYRU\CYRF\CYRN"}
3280 ⟨utf8⟩MACRO {ufn}   {"УФН"}
3281 MACRO {pu}   {"Phys.--Uspekhi"}
3282 MACRO {sjot} {"Sov.\ J.\ Opt.\ Technol."}
3283 MACRO {sjqe} {"Sov.\ J.\ Quantum Electron."}
3284 MACRO {sleb} {"Sov.\ Phys.--Leb.\ Inst.\ Rep."}
3285 MACRO {stph} {"Sov.\ Phys.--Techn.\ Phys."}
3286 MACRO {stphl}{"Sov.\ Techn.\ Phys.\ Lett."}
```

6.17   Main cycle

```
3287
3288 READ
3289
```

6.18   Sorting

Next chunk of code governs sorting reference list by authors' names and titles.

```
3290 ⟨∗sort | natbib⟩
```

```
3291 FUNCTION {sortify}
3292 { purify$
3293 ⟨!utf8⟩   "l" change.case$
3294 }
3295 ⟨/sort | natbib⟩
```

```
3296 ⟨∗sort⟩
3297 %% ====================================
3298 %% This version from old Gost package
3299 %%<*!natbib>
3300 FUNCTION {sort.format.names}
3301 { 's :=
3302   #1 'nameptr :=
3303   ""
3304   s num.names$ 'numnames :=
3305   numnames 'namesleft :=
3306     { namesleft #0 > }
3307     { nameptr #1 >
3308         { "   " * }
3309         'skip$
3310       if$
3311       s nameptr
3312       "{vv{ } }{ll{ }}{  f{ }}{  jj{ }}"
3313       format.name$ 't :=
3314       nameptr numnames = t "others" = and
3315         { "et al" * }
3316         %{ bbl.etal * }
3317         { t sortify * }
3318       if$
3319       nameptr #1 + 'nameptr :=
3320       namesleft #1 - 'namesleft :=
3321     }
3322   while$
3323 }
3324 %%</!natbib>
3325 %% This version from plainnat.bst
3326 %% It ignores second and subsequent authors but include year.
3327 %%<*natbib>
3328 %FUNCTION {sort.format.names}
3329 %{ 's :=
3330 %  #1 'nameptr :=
3331 %  ""
3332 %  s num.names$ 'numnames :=
3333 %  numnames 'namesleft :=
3334 %    { namesleft #0 > }
3335 %    {
3336 %      s nameptr "{vv{ } }{ll{ }}{  ff{ }}{  jj{ }}" format.name$ 't :=
3337 %      nameptr #1 >
3338 %        {
```

84

```
3339 %                "   "  *
3340 %              namesleft #1 = t "others" = and
3341 %                { "zzzzz" * }
3342 %                { numnames #2 > nameptr #2 = and
3343 %                    { "zz" * year field.or.null * "    " * }
3344 %                    'skip$
3345 %                  if$
3346 %                  t sortify *
3347 %                }
3348 %              if$
3349 %          }
3350 %          { t sortify * }
3351 %      if$
3352 %      nameptr #1 + 'nameptr :=
3353 %      namesleft #1 - 'namesleft :=
3354 %    }
3355 %  while$
3356 %}
3357 %%</natbib>
3358 %% ====================================
3359
```

```
3360 FUNCTION {sort.format.title}
3361 { 't :=
3362   "A " #2
3363     "An " #3
3364       "The " #4 t chop.word % Removes "The " if any
3365     chop.word                % Removes "An " if any
3366   chop.word                  % Removes "A " if any
3367   sortify
3368   #1 global.max$ substring$
3369 }
3370
```

```
3371 %% ====================================
3372 %% This version from old gost package.
3373 %%
3374 ⟨∗!natbib⟩
3375 FUNCTION {author.sort}
3376 { author empty$
3377     { key empty$
3378         { "to sort, need author or key in " cite$ * warning$
3379           ""
3380         }
3381         { key sortify }
3382       if$
3383     }
3384     {
3385       author num.names$ #4 <
```

```
3386          {author sort.format.names }
3387          {title sort.format.title}
3388        if$
3389      }
3390    if$
3391 }
```
3392 ⟨/!natbib⟩
```
3393 %% This version from plainnat.bst
```
3394 ⟨∗natbib⟩
```
3395 FUNCTION {author.sort}
3396 { author empty$
3397     { key empty$
3398         { "to sort, need author or key in " cite$ * warning$
3399           ""
3400         }
3401         { key sortify }
3402       if$
3403     }
3404     { author sort.format.names }
3405   if$
3406 }
```
3407 ⟨/natbib⟩
```
3408 %% =====================================
3409
```

author.title.sort   The function `author.title.sort` is used in the `presort` function only.

```
3410 FUNCTION {author.title.sort}
3411 { author empty$
3412     { title empty$
3413         { key empty$
3414             { "to sort, need author, title, or key in " cite$ * warning$
3415               ""
3416             }
3417             { key sortify }
3418           if$
3419         }
3420         { title sort.format.title }
3421       if$
3422     }
3423     {
3424       author num.names$ #4 <
3425         {author sort.format.names }
3426         {title sort.format.title}
3427       if$
3428     }
3429   if$
3430 }
```

author.editor.sort   The function `author.editor.sort`, `author.organization.sort`, `editor.organization.sort`
organization.sort   are not currently used. See commented text in function `presort`#2. ⟨NB: WE NEED TO CORRECT
organization.sort   THAT!⟩

86

```
3431 ⟨∗natbib⟩
3432 %FUNCTION {author.editor.sort}
3433 %{ author empty$
3434 %    { editor empty$
3435 %        { key empty$
3436 %            { "to sort, need author, editor, or key in " cite$ * warning$
3437 %              ""
3438 %            }
3439 %            { key sortify }
3440 %          if$
3441 %        }
3442 %        { editor sort.format.names }
3443 %      if$
3444 %    }
3445 %    { author sort.format.names }
3446 %  if$
3447 %}
3448 %
3449 %FUNCTION {author.organization.sort}
3450 %{ author empty$
3451 %    { organization empty$
3452 %        { key empty$
3453 %            { "to sort, need author, organization, or key in " cite$ * warning$
3454 %              ""
3455 %            }
3456 %            { key sortify }
3457 %          if$
3458 %        }
3459 %        { "The " #4 organization chop.word sortify }
3460 %      if$
3461 %    }
3462 %    { author sort.format.names }
3463 %  if$
3464 %}
3465
3466 FUNCTION {editor.organization.sort}
3467 { editor empty$
3468     { organization empty$
3469         { key empty$
3470             { "to sort, need editor, organization, or key in " cite$ * warning$
3471               ""
3472             }
3473             { key sortify }
3474           if$
3475         }
3476         { "The " #4 organization chop.word sortify }
3477       if$
3478     }
3479     { editor sort.format.names }
3480   if$
```

```
3481 }
3482 ⟨/natbib⟩
```

presort   Function to compute `sort.key$`. What is the space string `"␣␣␣"` for? Version #1 is for 'sort'
and '!natbib' options. Version #2 is for 'sort' and 'natbib' options. Version #3 is for '!sort' and
'natbib' options.

```
3483 ⟨∗!natbib⟩
3484 FUNCTION {presort}%#1
3485 {
3486   author.title.sort
3487   "     "
3488   *
3489   year field.or.null sortify
3490   *
3491   "     "
3492   *
3493   title field.or.null
3494   sort.format.title
3495   *
3496   #1 entry.max$ substring$
3497   'sort.key$ :=
3498 }
3499 ⟨/!natbib⟩
3500 ⟨∗natbib⟩
3501 FUNCTION {presort}%#2
3502 { calc.label
3503   label sortify
3504   %author.title.sort
3505   "     "
3506   *
3507   % ========= plainnat.bst =========
3508 %  type$ "book" =
3509 %  type$ "inbook" =
3510 %  or
3511 %    'author.editor.sort
3512 %    { type$ "proceedings" =
3513 %       'editor.organization.sort
3514 %       { type$ "manual" =
3515 %          'author.organization.sort
3516 %          'author.sort
3517 %        if$
3518 %       }
3519 %     if$
3520 %    }
3521 %  if$
3522   author.title.sort
3523   "     "
3524   *
3525   year field.or.null sortify
3526   *
3527   "     "
```

88

```
3528   *
3529   %cite$
3530   title field.or.null sort.format.title
3531   *
3532   #1 entry.max$ substring$
3533   'sort.label :=
3534   sort.label *
3535   % ==============================
3536   #1 entry.max$ substring$
3537   'sort.key$ :=
3538 }
```
⟨/natbib⟩
⟨/sort⟩

⟨∗!sort⟩
⟨∗natbib⟩
```
INTEGERS { seq.num }

FUNCTION {init.seq}
{ #0 'seq.num :=}

EXECUTE {init.seq}

FUNCTION {int.to.fix}
{ "000000000" swap$ int.to.str$ *
  #-1 #10 substring$
}

FUNCTION {presort}%#3
{
  calc.label              % computes label
  label sortify           % initiates sort.label
  "    "
  *
  seq.num #1 + 'seq.num := % advance seq.num
  seq.num  int.to.fix      % prepend seq.num with 0s
  'sort.label :=           % set sort.label to seq.num
  sort.label *             % append seq.num to label
  #1 entry.max$ substring$ % cut if too long
  'sort.key$ :=            % set sort.key$
}
```
⟨/natbib⟩
⟨/!sort⟩

⟨∗sort | natbib⟩
```
ITERATE {presort}

SORT
```
⟨/sort | natbib⟩

6.19   Bibliography list

We need to find longest label to put in into the argument of the `thebibliography` environment. In case of `natbib` options we also need to compute extra suffix for the `year` field if there two or more entries for given label (=author/editor/organization) in that year.

Declare global (external) strings used in calculation of the longest label.

```
3579 ⟨!natbib⟩STRINGS { longest.label }
3580 ⟨natbib⟩STRINGS { longest.label last.label next.extra }
3581
3582 ⟨!natbib⟩INTEGERS { number.label longest.label.width }
3583 ⟨natbib⟩INTEGERS { number.label longest.label.width last.extra.num  }
3584
```

`ize.longest.label`   Initialize those string.

```
3585 ⟨∗!natbib⟩
3586 FUNCTION {initialize.longest.label}
3587 { "" 'longest.label :=
3588   #1 'number.label :=
3589   #0 'longest.label.width :=
3590 }
3591 ⟨/!natbib⟩
3592 ⟨∗natbib⟩
3593 FUNCTION {initialize.longest.label}
3594 { "" 'longest.label :=
3595   #0 int.to.chr$ 'last.label :=
3596   "" 'next.extra :=
3597   #0 'longest.label.width :=
3598   #0 'last.extra.num :=
3599   #0 'number.label :=
3600 }
3601 ⟨/natbib⟩
3602
3603 EXECUTE {initialize.longest.label}
3604
```

`ize.longest.label`   Iterate though the list of entries to compute `label`.

```
3605 ⟨∗!natbib⟩
3606 FUNCTION {forward.pass}
3607 { number.label int.to.str$ 'label :=
3608   number.label #1 + 'number.label :=
3609   label width$ longest.label.width >
3610     { label 'longest.label :=
3611       label width$ 'longest.label.width :=
3612     }
3613     'skip$
3614   if$
3615 }
3616 ⟨/!natbib⟩
```

```
3617 ⟨∗natbib⟩
3618 FUNCTION {forward.pass}
3619 { last.label label =
3620     { last.extra.num #1 + 'last.extra.num :=
3621       last.extra.num int.to.chr$ 'extra.label :=
3622     }
3623     { "a" chr.to.int$ 'last.extra.num :=
3624       "" 'extra.label :=
3625       label 'last.label :=
3626     }
3627   if$
3628   number.label #1 + 'number.label :=
3629 }
3630 ⟨/natbib⟩
3631
3632 ITERATE {forward.pass}
3633
```

reverse.pass   Natbib styles require reverse iteration over all entries.

```
3634 ⟨∗natbib⟩
3635 FUNCTION {reverse.pass}
3636 { next.extra "b" =
3637     { "a" 'extra.label := }
3638     'skip$
3639   if$
3640   extra.label 'next.extra :=
3641   extra.label
3642   duplicate$ empty$
3643     'skip$
3644     { "{\natexlab{" swap$ * "}}" * }
3645   if$
3646   'extra.label :=
3647   %%%label extra.label * 'label :=
3648 }
3649
3650 REVERSE {reverse.pass}
3651
3652 FUNCTION {bib.sort.order}
3653 { sort.label   'sort.key$ :=
3654 }
3655
3656 ITERATE {bib.sort.order}
3657
3658 SORT
3659 ⟨/natbib⟩
3660
```

begin.bib   Within thebibliography environment we define few formatting macros for user to customize how the reference list is formatted.

```
3661 FUNCTION {begin.bib}
3662 { "\begin{thebibliography}{"  longest.label  * "}" * write$ newline$
```

```
3663    "\def\selectlanguageifdefined#1{"                   write$ newline$
3664    "\expandafter\ifx\csname date#1\endcsname\relax"    write$ newline$
3665 %  "\else\language\csname l@#1\endcsname\fi}"           write$ newline$
3666    "\else\selectlanguage{#1}\fi}"                       write$ newline$
3667    "\providecommand*{\href}[2]{{\small #2}}"            write$ newline$
3668    "\providecommand*{\url}[1]{{\small #1}}"             write$ newline$
3669    "\providecommand*{\BibUrl}[1]{\url{#1}}"             write$ newline$
3670    "\providecommand{\BibAnnote}[1]{}"                   write$ newline$
3671    "\providecommand*{\BibEmph}[1]{#1}"                  write$ newline$
3672    "\ProvideTextCommandDefault{\cyrdash}{\iflanguage{russian}{\hbox to.8em{--\hss--}}{\textemdash}}" write$ ne
3673    "\providecommand*{\BibDash}{\ifdim\lastskip>0pt\unskip\nobreak\hskip.2em plus 0.1em\fi" write$ newline$
3674    "\cyrdash\hskip.2em plus 0.1em\ignorespaces}"        write$ newline$
3675    "\renewcommand{\newblock}{\ignorespaces}"            write$ newline$
3676 ⟨natbib⟩  "\providecommand{\natexlab}[1]{#1}" write$ newline$
3677    preamble$ empty$
3678      'skip$
3679      { preamble$ write$ newline$ }
3680    if$
3681 }
3682
3683
3684 EXECUTE {begin.bib}
3685
3686 EXECUTE {init.state.consts}
3687
3688 ITERATE {call.type$}
3689
```

end.bib

```
3690 FUNCTION {end.bib}
3691 { newline$
3692 %  "\catcode`\/=11"         write$ newline$
3693    "\end{thebibliography}" write$ newline$
3694 }
3695
3696 EXECUTE {end.bib}
3697
3698 ⟨/bst⟩
```

7 Change History

8   Index

Numbers written in dark blue refer to the page where the corresponding entry is described; numbers in black roman refer to the code lines where the entry is used.