numberpt: counters spelled out in Portuguese

Miguel Vinícius Santini Frasson

2019–07–13 version 1.0

Contents

1 Introduction

The package `numberpt` provides a counter style like `\arabic`, `\alph` etc., but that spells numbers in Portuguese, like "um" (one in Portuguese), "dois" (two), "três" (three), ..., up to "novecentos e noventa e nove mil novecentos e noventa e nove" (999,999). There are counter commands to output the text in "lowercase", "First word capitalized", "All Words Capitalized" ou in "UPPERCASE".

2 User commands

\numberpt
\Numberpt
\NumberPt
\NUMBERPT

Macros to output counters in Portuguese in "lowercase", "Capitalized" or in "UPPERCASE"—:

\numberpt{*counter*} (ex: vinte e três),
\Numberpt{*counter*} (ex: Vinte e três),
\NumberPt{*counter*} (ex: Vinte E Três),
\NUMBERPT{*counter*} (ex: VINTE E TRÊS).

Example:

```
\renewcommand{\thechapter}{\Numberpt{chapter}}
```

will produce chapter headers like:

Capítulo Um
Capítulo Dois

By default, command \NumberPt for all capitalized numbers output connectors "e" in uppercase. For example, if page counter is 23, then \NumberPt{page} → Vinte E Três. To change this behavior (connector "e" in lowercase), use starred versions \NumberPt*.

3 Package options

catorze
quatorze

For 14, both forms "catorze" e "quatorze" are correct. Use package option catorze (default) or quatorze to select which form is used.

dezesseis
dezasseis

In Brazil, the forms for 16, 17 and 19 are "dezesseis", "dezessete" and "dezenove". In Portugal, the forms are "dezasseis", "dezassete" and "dezanove". To select Brazilian Portuguese forms (default), use option dezesseis; for European Portuguese form, use dezasseis.

4 Code

4.1 Package identification

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{numberpt}[2019/07/12 v1.0 Counters as numbers in Portuguese]
```

We use LaTeX3 in this package.

```
\RequirePackage{expl3,xparse}
\ExplSyntaxOn
```

4.2 Package options

Package options declarations catorze or quatorze, and dezasseis or dezesseis. Creating undocumented "user commands" \NumberPTcatorze, \NumberPTquatorze, \NumberPTdezasseis and \NumberPTdezasseis that set switches.

```
\bool_new:N \NumberPT_catorze
\bool_new:N \NumberPT_dezesseis
```

```
\newcommand{\NumberPTcatorze}{\bool_set_true:N  \NumberPT_catorze}
\newcommand{\NumberPTquatorze}{\bool_set_false:N  \NumberPT_catorze}
\DeclareOption{ catorze  }{ \NumberPTcatorze }
\DeclareOption{ quatorze }{ \NumberPTquatorze }
\ExecuteOptions{ catorze }
\newcommand{\NumberPTdezesseis}{\bool_set_true:N  \NumberPT_dezesseis}
\newcommand{\NumberPTdezasseis}{\bool_set_false:N  \NumberPT_dezesseis}
\DeclareOption{ dezasseis  }{ \NumberPTdezasseis}
\DeclareOption{ dezesseis  }{ \NumberPTdezesseis}
\ExecuteOptions{ dezesseis }
\ProcessOptions
```

4.3 Case handling

Command `\NumberPT_case:nn` and switches govern capitalization.

```
\bool_new:N \NumberPT_capital
\bool_new:N \NumberPT_capitalfirst
\bool_new:N \NumberPT_capitale
\bool_new:N \NumberPT_uppercase
\cs_new:Nn \NumberPT_case:nn {
  \bool_if:NTF \NumberPT_capital {
    \uppercase { #1 }
  } {
    #1
  }
  \bool_if:NTF \NumberPT_uppercase {
    \uppercase { #2 }
  } {
    #2 }
  \bool_if:NT \NumberPT_capitalfirst {
    \bool_set_false:N \NumberPT_capital
  }
}
```

4.4 Output for numbers with 1, 2 or 3 digits

`\NumberPT_u:n{`U`}` takes digit U and outputs “zero”, “um”, “dois”, . . . , “nove”.

```
\cs_new:Nn \NumberPT_u:n {
  \if_case:w #1
    \NumberPT_case:nn {z}{ero}\or:
    \NumberPT_case:nn {u}{m}\or:
    \NumberPT_case:nn {d}{ois}\or:
    \NumberPT_case:nn {t}{r\^es}\or:
    \NumberPT_case:nn {q}{uatro}\or:
    \NumberPT_case:nn {c}{inco}\or:
    \NumberPT_case:nn {s}{eis}\or:
    \NumberPT_case:nn {s}{ete}\or:
    \NumberPT_case:nn {o}{ito}\or:
```

```
    \NumberPT_case:nn {n}{ove}%
  \fi:}
```

Command `\NumberPT_e_u:n{`U`}` outputs " e um" etc, if $U > 0$, otherwise outputs nothing.

```
\cs_new:Nn \NumberPT_e_u:n {
  \int_compare:nNnT #1 > 0 {
    \space \bool_if:nTF \NumberPT_capitale {E}{e} \space \NumberPT_u:n #1
  }
}
```

Commands `\NumberPT_du:nn{`D`}{`U`}` and `\NumberPT_e_du:nn{`D`}{`U`}` are similar, with 2 digits. Numbers from 0 to 19 are all special cases.

```
\cs_new:Nn \NumberPT_du:nn {
  \if_case:w #1 % #1=0
    \NumberPT_u:n #2
  \or: % #1=1
    \if_case:w #2
      \NumberPT_case:nn {d}{ez}\or:
      \NumberPT_case:nn {o}{nze}\or:
      \NumberPT_case:nn {d}{oze}\or:
      \NumberPT_case:nn {t}{reze}\or:
      \bool_if:NTF \NumberPT_catorze {
        \NumberPT_case:nn {c}{atorze}
      } {
        \NumberPT_case:nn {q}{uatorze}
      }\or:
      \NumberPT_case:nn {q}{uinze}\or:
      \bool_if:NTF \NumberPT_dezesseis {
        \NumberPT_case:nn {d}{ezesseis}
      } {
        \NumberPT_case:nn {d}{ezasseis}
      }\or:
      \bool_if:NTF \NumberPT_dezesseis {
        \NumberPT_case:nn {d}{ezessete}
      } {
        \NumberPT_case:nn {d}{ezassete}
      }\or:
      \NumberPT_case:nn {d}{ezoito}\or:
      \bool_if:NTF \NumberPT_dezesseis {
        \NumberPT_case:nn {d}{ezenove}
      } {
        \NumberPT_case:nn {d}{ezanove}
      }\or:
    \fi:
  \or: % #1>1
    \NumberPT_case:nn {v}{inte} \NumberPT_e_u:n #2
  \or:
    \NumberPT_case:nn {t}{rinta} \NumberPT_e_u:n #2
  \or:
```

```
    \NumberPT_case:nn {q}{uarenta} \NumberPT_e_u:n #2
  \or:
    \NumberPT_case:nn {c}{inquenta} \NumberPT_e_u:n #2
  \or:
    \NumberPT_case:nn {s}{essenta} \NumberPT_e_u:n #2
  \or:
    \NumberPT_case:nn {s}{etenta} \NumberPT_e_u:n #2
  \or:
    \NumberPT_case:nn {o}{itenta} \NumberPT_e_u:n #2
  \or:
    \NumberPT_case:nn {n}{oventa} \NumberPT_e_u:n #2
  \fi:
}

\cs_new:Nn \NumberPT_e_du:nn {
  \int_compare:nNnT { #1 + #2 } > 0 {
    \space \bool_if:nTF \NumberPT_capitale {E}{e} \space \NumberPT_du:nn #1 #2
  }
}
```

`\NumberPT_cdu:nnn{`C`}{`D`}{`U`}` takes digits C, D and U and outputs correspondent number.

```
\cs_new:Nn \NumberPT_cdu:nnn {%
  \if_case:w #1 % #1=0
    \NumberPT_du:nn #2 #3
  \or: % #1=1
    \int_compare:nNnTF { #2 + #3 } = 0  % "cem" if 00 or "cento e " + finish
      { \NumberPT_case:nn {c}{em} }
      { \NumberPT_case:nn {c}{ento} \NumberPT_e_du:nn #2 #3 }
  \or: % #1>1
    \NumberPT_case:nn {d}{uzentos}  \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {t}{rezentos} \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {q}{uatrocentos} \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {q}{uinhentos} \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {s}{eiscentos} \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {s}{etecentos} \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {o}{itocentos} \NumberPT_e_du:nn #2 #3
  \or:
    \NumberPT_case:nn {n}{ovecentos} \NumberPT_e_du:nn #2 #3
  \fi:
}
```

4.5 Output for numbers with 4, 5 or 6 digits

If a number $abcdef$ has between 4 and 6 digits, the rule for spelling this number is[1]:

- spell the 3-digits number abc (unless $abc = 001$) + "mil"
- stop if $def = 000$, otherwise:
- add the connector "e" for numbers
  - finishing in 00, *i.e.*, $e + f = 0$
  - without digit for hundreds , *i.e.*, $d = 0$
- spell the 3-digits number def

```
\cs_new:Nn \NumberPT_e_cdu:nnn {
  \int_compare:nNnF {#1 + #2 + #3} = 0 {
    \space
    \bool_if:nT { \int_compare_p:nNn #1 = 0 || \int_compare_p:nNn {#2 + #3} = 0 }
    {
      \bool_if:nTF \NumberPT_capitale {E}{e}
      \space
    }
    \NumberPT_cdu:nnn #1 #2 #3
  }
}
```

Output for numbers with up to 6 digits.

```
\cs_new:Nn \NumberPT_abcdef:nnnnnn {
  \int_compare:nNnTF {#1 + #2 + #3} = 0 %abc=000
  {
    \NumberPT_cdu:nnn #4 #5 #6
  }
  % else
  {
    % avoid "um mil", using just "mil"
    \int_compare:nNnF {100 * #1 + 10 * #2 + #3} = 1 {
      \NumberPT_cdu:nnn #1 #2 #3
      \space
    }
    \NumberPT_case:nn {m}{il}
    \NumberPT_e_cdu:nnn #4 #5 #6
  }
}
\cs_generate_variant:Nn \NumberPT_abcdef:nnnnnn {xxxxxx}
```

[1] See www.languagesandnumbers.com/como-contar-em-portugues-brasil/pt/por-bra/

4.6 Conversion from counter to digits

Now, conversion from counter to digits. digits will be stored in macros `\NumberPT_digit_u`, `\NumberPT_digit_d`, `\NumberPT_digit_c`. Decomposition is done with mod.

```
\int_new:N \NumberPT_digit_u
\int_new:N \NumberPT_digit_d
\int_new:N \NumberPT_digit_c
\int_new:N \NumberPT_digit_um
\int_new:N \NumberPT_digit_dm
\int_new:N \NumberPT_digit_cm

\cs_new:Nn \NumberPT_decompose:n {
  \int_set:Nn \l_tmpa_int { #1 }
  \int_set:Nn \NumberPT_digit_u { \int_mod:nn { \l_tmpa_int } { 10 } }
  \int_set:Nn \l_tmpa_int { ( \l_tmpa_int - \NumberPT_digit_u ) / 10 }
  \int_set:Nn \NumberPT_digit_d { \int_mod:nn { \l_tmpa_int } { 10 } }
  \int_set:Nn \l_tmpa_int { ( \l_tmpa_int - \NumberPT_digit_d ) / 10 }
  \int_set:Nn \NumberPT_digit_c { \int_mod:nn { \l_tmpa_int } { 10 } }
  \int_set:Nn \l_tmpa_int { ( \l_tmpa_int - \NumberPT_digit_c ) / 10 }
  \int_set:Nn \NumberPT_digit_um { \int_mod:nn { \l_tmpa_int } { 10 } }
  \int_set:Nn \l_tmpa_int { ( \l_tmpa_int - \NumberPT_digit_um ) / 10 }
  \int_set:Nn \NumberPT_digit_dm { \int_mod:nn { \l_tmpa_int } { 10 } }
  \int_set:Nn \l_tmpa_int { ( \l_tmpa_int - \NumberPT_digit_dm ) / 10 }
  \int_set:Nn \NumberPT_digit_cm { \int_mod:nn { \l_tmpa_int } { 10 } }
}
```

Now, command that prints counter: `\NumberPT_print_counter:n`. Declaring error message in case counter is out of range from 0 to 999. Even in case of this error, falls back to arabic number.

```
\msg_new:nnn
{ numberpt } { counter-out-of-range } {Counter~‘#1’~out~of~range~0~...~999999}

\cs_new:Nn \NumberPT_print_counter:n {
  \int_set:Nn \l_tmpa_int { \value{#1} }
  \bool_if:nTF {
    \int_compare_p:n { \l_tmpa_int >= 0 } &&
    \int_compare_p:n { \l_tmpa_int < 1000000 }
  }
  {
    \NumberPT_decompose:n { \l_tmpa_int }
    \NumberPT_abcdef:xxxxxx
    { \int_use:N \NumberPT_digit_cm }
    { \int_use:N \NumberPT_digit_dm }
    { \int_use:N \NumberPT_digit_um }
    { \int_use:N \NumberPT_digit_c  }
    { \int_use:N \NumberPT_digit_d  }
    { \int_use:N \NumberPT_digit_u  }
  }
```

```
  {
    \msg_error:nnn { numberpt } { counter-out-of-range } {#1}
    \arabic{#1}
  }
}
```

4.7 User macros

Finally, user macros. Each macro sets switches for capitalization and calls `\NumberPT_print_counter:n`.

```
\NewDocumentCommand{\numberpt}{m}{
  \bool_set_false:N \NumberPT_capital
  \bool_set_false:N \NumberPT_capitale
  \bool_set_false:N \NumberPT_uppercase
  \NumberPT_print_counter:n { #1 }
}

\NewDocumentCommand{\NumberPt}{s m}{
  \bool_set_true:N \NumberPT_capital
  \bool_set:Nn \NumberPT_capitale {! #1}
  \bool_set_false:N \NumberPT_capitalfirst
  \bool_set_false:N \NumberPT_uppercase
  \NumberPT_print_counter:n { #2 }
}

\NewDocumentCommand{\Numberpt}{m}{
  \bool_set_true:N \NumberPT_capital
  \bool_set_false:N \NumberPT_capitale
  \bool_set_true:N \NumberPT_capitalfirst
  \bool_set_false:N \NumberPT_uppercase
  \NumberPT_print_counter:n { #1 }
}

\NewDocumentCommand{\NUMBERPT}{m}{
  \bool_set_true:N \NumberPT_capital
  \bool_set_true:N \NumberPT_capitale
  \bool_set_true:N \NumberPT_uppercase
  \NumberPT_print_counter:n { #1 }
}
```

End of code.

```
\ExplSyntaxOff
\endinput
```

5 Tests

Here we create a counter and spell the numbers for some selected values.

5.1 Tests with `\numberpt`

1: um
2: dois
3: três
4: quatro
5: cinco
6: seis
7: sete
8: oito
9: nove
10: dez
11: onze
12: doze
13: treze
14: catorze
15: quinze
16: dezesseis
17: dezessete
18: dezoito
19: dezenove
20: vinte
21: vinte e um
22: vinte e dois
30: trinta
31: trinta e um
40: quarenta
41: quarenta e um
50: cinquenta
51: cinquenta e um
60: sessenta
61: sessenta e um
70: setenta
71: setenta e um
80: oitenta
81: oitenta e um
90: noventa
91: noventa e um
100: cem
101: cento e um

5.2 Tests with `\Numberpt`

199: Cento e noventa e nove
200: Duzentos
201: Duzentos e um
300: Trezentos
301: Trezentos e um
400: Quatrocentos
401: Quatrocentos e um
500: Quinhentos
501: Quinhentos e um

5.3 Tests with `\NumberPt`

600: Seiscentos
601: Seiscentos E Um
700: Setecentos
701: Setecentos E Um

5.4 Tests with `\NumberPt*`

800: Oitocentos
801: Oitocentos e Um
900: Novecentos
999: Novecentos e Noventa e Nove

5.5 Tests with `\NUMBERPT`

1000: MIL
1001: MIL E UM
1099: MIL E NOVENTA E NOVE
1100: MIL E CEM
1101: MIL CENTO E UM
2000: DOIS MIL
2001: DOIS MIL E UM

2099: DOIS MIL E NOVENTA E NOVE
2100: DOIS MIL E CEM
2101: DOIS MIL CENTO E UM
2200: DOIS MIL E DUZENTOS
2201: DOIS MIL DUZENTOS E UM
2299: DOIS MIL DUZENTOS E NOVENTA E NOVE
2300: DOIS MIL E TREZENTOS

5.6 Tests with 14, 16, 17, 19

14: catorze (must be “catorze”)

14: quatorze (must be “quatorze”)

16: dezasseis (must be “dezasseis”)
17: dezassete (must be “dezassete”)
19: dezanove (must be “dezanove”)

16: dezesseis (must be “dezesseis”)
17: dezessete (must be “dezessete”)
19: dezenove (must be “dezenove”)