NAME

Sort::Key::Natural – fast natural sorting

SYNOPSIS

```
    use Sort::Key::Natural qw(natsort);

    my @data = qw(foo1 foo23 foo6 bar12 bar1
                  foo bar2 bar-45 foomatic b-a-r-45);

    my @sorted = natsort @data;

    print "@sorted\n";
    # prints:
    #   b-a-r-45 bar1 bar2 bar12 bar-45 foo foo1 foo6 foo23 foomatic

    use Sort::Key::Natural qw(natkeysort);

    my @objects = (...);
    my @sorted = natkeysort { $_->get_id } @objects;
```

DESCRIPTION

This module extends the Sort::Key family of modules to support natural sorting.

Under natural sorting, strings are split at word and number boundaries, and the resulting substrings are compared as follows:

- numeric substrings are compared numerically
- alphabetic substrings are compared lexically
- numeric substrings come always before alphabetic substrings

Spaces, symbols and non-printable characters are only considered for splitting the string into its parts but not for sorting. For instance `foo-bar-42` is broken in three substrings `foo`, `bar` and `42` and after that the dashes are ignored.

Note, that the sorting is case sensitive. To do a case insensitive sort you have to convert the keys explicitly:

```
  my @sorted = natkeysort { lc $_ } @data
```

Also, once this module is loaded, the new type `natural` (or `nat`) will be available from Sort::Key::Maker. For instance:

```
  use Sort::Key::Natural;
  use Sort::Key::Maker i_rnat_keysort => qw(integer -natural);
```

creates a multi-key sorter `i_rnat_keysort` accepting two keys, the first to be compared as an integer and the second in natural descending order.

There is also an alternative set of natural sorting functions that recognize floating point numbers. They use the key type `natwf` (abbreviation of `natural_with_floats`).

FUNCTIONS

the functions that can be imported from this module are:

natsort `@data`
: returns the elements of `@data` sorted in natural order.

rnatsort `@data`
: returns the elements of `@data` sorted in natural descending order.

natkeysort { CALC_KEY($_) } `@data`
: returns the elements on `@array` naturally sorted by the keys resulting from applying them `CALC_KEY`.

rnatkeysort { CALC_KEY($_) } `@data`
: is similar to `natkeysort` but sorts the elements in descending order.

natsort_inplace `@data`
rnatsort_inplace `@data`
natkeysort_inplace { CALC_KEY($_) } `@data`
rnatkeysort_inplace { CALC_KEY($_) } `@data`
: these functions are similar respectively to `natsort`, `rnatsort`, `natsortkey` and `rnatsortkey`, but they sort the array `@data` in place.

`$key` = mkkey_natural `$string`
: given `$string`, returns a key that can be compared lexicographically to another key obtained in the same manner, results in the same order as comparing the former strings as in the natural order.

: If the argument `$key` is not provided it defaults to `$_`.

natwfsort `@data`
rnatwfsort `@data`
natwfkeysort { CALC_KEY($_) } `@data`
rnatwfkeysort { CALC_KEY($_) } `@data`
natwfsort_inplace `@data`
rnatwfsort_inplace `@data`
natwfkeysort_inplace { CALC_KEY($_) } `@data`
rnatwfkeysort_inplace { CALC_KEY($_) } `@data`
mkkey_natural_with_floats `$key`
: this ugly named set of functions perform in the same way as its s/natwf/nat/ counterpart with the difference that they honor floating point numbers embedded inside the strings.

: In this context a floating point number is a string matching the regular expression `/[+\-]?\d+(\.\d*)?/`. Note that numbers with an exponent part (i.e. `1.12E-12`) are not recognized as such.

: Note also that numbers without an integer part (i.e. `.2` or `-.12`) are not supported either.

SEE ALSO

Sort::Key, Sort::Key::Maker.

Other module providing similar functionality is Sort::Naturally.

COPYRIGHT AND LICENSE