The erw-l3 package*

January 28, 2022

Abstract

Utilities based on LaTeX3[1], such as `\erw_merge_sort:nNn`.

Contents

---

*This file describes version v4.2, last revised 2022-01-28.

Part I
Usage

1 boilerplate

```
\erw_keys_set:n
\erw_keys_set:nn
```

```
\erw_keys_set:n{⟨keyval list⟩}
```

```
\erw_identity:n             ⋆
\erw_int_incr:n             ⋆
\erw_swap:nn                ⋆
\erw_swap:ne                ⋆
\erw_name_signature_cs:N    ⋆
```

2 quark

```
\erw_all_q:w
\erw_remove_first_q:w
\erw_first_q:w
\erw_remove_last_q:w
\erw_last_q:w
```

```
\erw_remove_last_q:w⟨tokenlist⟩ \q_recursion_tail\q_recursion_stop
```

3 predicate

```
new_compare_p
```

```
\erw_keys_set:n{ new_compare_p = {⟨name⟩}{⟨signature⟩}{⟨predicate⟩} }
```

Instance

erw_compare_p:nNnNn

erw_int_incr_p:nn

4 op's on lists

```
\erw_remove_first:n
\erw_remove_last:n
\erw_first:n
\erw_last:n
\erw_adjacent_insert:nn
\erw_adjacent_insert:en
```

5 algo

```
\erw_split_even:n
\erw_split_even:e
\erw_merge_sort:nNn
\thread_sort:nnNn
\erw_filter_uniq:nn
\erw_filter_uniq:n
```

```
\erw_thread_sort:nnNn{⟨first sorted list⟩}{⟨second sorted list⟩}{⟨compare
predicate name⟩}<|>
\erw_merge_sort:nNn{⟨compare predicate name⟩}<|>{⟨unsorted list⟩}
\erw_filter_uniq:nn{⟨compare predicate⟩}{⟨tokenlist⟩}
\erw_filter_uniq:n{⟨ascending intergers⟩}
```

6 code

```
\erw_parameter:n
\erw_parameter:nn
\argument:nn
```

```
\erw_parameter:n{⟨arity⟩}
\erw_parameter:nn{⟨start pos⟩}{⟨arity⟩}
\erw_argument:nn{⟨start pos⟩}{⟨signature⟩}
```

Part II
Other

1 Bibliograhy

[1] The LaTeX3 Project Team. *The LaTeX3 interfaces*. `https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/expl3.pdf`. 2019.

2 Support

This package is available from `https://github.com/rogard/erw-l3`.

Part III
Implementation

```
⟨*package⟩
⟨@@=erw⟩
%      \ExplSyntaxOn
```

1 kernel

```
\cs_generate_variant:Nn\int_compare_p:nNn{eNe}
\cs_generate_variant:Nn\int_eval:n{e}
\cs_generate_variant:Nn\prg_new_conditional:Nnn{c}
\cs_generate_variant:Nn\prg_replicate:nn{e}
\cs_generate_variant:Nn\regex_gset:Nn{c}
\cs_generate_variant:Nn\regex_log:N{c}
\cs_generate_variant:Nn\regex_match:NnTF{c}
\cs_generate_variant:Nn\tl_to_str:n{e}
\cs_generate_variant:Nn\prop_put:Nnn{Nne}
```

2 boilerplate

```
\msg_new:nnnn{__erw}{text}{text~is~not~loaded}{load~amsmath}
\cs_new:Npn \__erw_text:n #1
{\cs_if_exist:NTF\text{\text{#1}}{\msg_error:nn{__erw}{text}}}
\cs_new:Npn\__erw_empty:w #1 \q_recursion_stop {\c_empty_tl}
\cs_new_protected:Nn\erw_keys_set:n{ \keys_set:nn{__erw}{#1} }
\cs_new_protected:Nn\erw_keys_set:nn{ \keys_set:nn{__erw / #1}{#2} }
\cs_generate_variant:Nn\erw_apply:Nw{c}
\cs_new:Npn \erw_identity:n#1{#1}
\cs_new:Npn \erw_int_incr:n#1{\int_eval:n{#1+1}}
\cs_new:Npn \erw_swap:nn#1#2{#2#1}
\cs_generate_variant:Nn \erw_swap:nn{e}
\cs_new:Npn \erw_name_signature_cs:N #1
{ \exp_last_unbraced:Ne
  \__erw_name_signature_cs:nnn{\cs_split_function:N#1}}
\cs_new:Nn \__erw_name_signature_cs:nnn{{#1}{#2}}
```

3 quark

```
\msg_new:nnn{erw}{quark-only-tail}
{requires~tail;~got~’#1’;~\msg_line_context:}
\cs_new:Npn
\erw_all_q:w
#1
\q_recursion_stop
{%
  \erw_remove_last_q:w#1\q_recursion_stop
  \erw_last_q:w#1\q_recursion_stop
}
\cs_new:Npn
\erw_remove_first_q:w
#1 % <tokenlist ending with recursion tail>
\q_recursion_stop
{\quark_if_recursion_tail_stop:n{#1}
  \__erw_remove_first_q:nw#1\q_recursion_stop}
\cs_new:Npn
\__erw_remove_first_q:nw
```

```
#1 % <head>
#2 % <rest>
\q_recursion_stop
{\erw_remove_last_q:w#2\q_recursion_stop
  \erw_last_q:w#2\q_recursion_stop}
\cs_new:Npn
\erw_first_q:w
#1
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#1}
  \__erw_first_q:enw{ \tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop }
\cs_new:Npn
\__erw_first_q:nnw
#1 % <head is group>
#2 % <head>
#3 % <rest>
\q_recursion_stop
{%
  \bool_if:nTF{#1}{{#2}}{#2}
}
\cs_generate_variant:Nn\__erw_first_q:nnw{e}
\cs_new:Npn
\erw_remove_last_q:w #1 \q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#1}
  \__erw_remove_last_q:ew{\tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop }
\cs_new:Npn
\__erw_remove_last_q:nw
#1 % <head is group>
#2 % <tokenlist>
\q_recursion_stop
{ \__erw_remove_last_q:nnw{#1}#2\q_recursion_stop }
\cs_generate_variant:Nn\__erw_remove_last_q:nw{e}
\cs_new:Npn
\__erw_remove_last_q:nnw
#1 % <head is group>
#2 % <head>
#3 % <rest>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#3}
  \bool_if:nTF{#1}{{#2}}{#2}
  \__erw_remove_last_q:ew {\tl_if_head_is_group_p:n{#3}} #3 \q_recursion_stop
}
\cs_generate_variant:Nn\__erw_remove_last_q:nnw{e}
\cs_new:Npn
\erw_last_q:w #1 \q_recursion_stop
{\quark_if_recursion_tail_stop:n{#1}
  \__erw_last_q:ew{\tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop}
\cs_new:Npn
\__erw_last_q:nw
#1 % <head is group>
#2 % <tokenlist>
```

```
\q_recursion_stop
{ \__erw_last_q:nnw{#1}#2\q_recursion_stop }
\cs_generate_variant:Nn\__erw_last_q:nw{e}
\cs_new:Npn
\__erw_last_q:nnw
#1 % <head is group>
#2 % <head>
#3 % <rest>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#3}{ \bool_if:nTF{#1}{{#2}}{#2} }
  \__erw_last_q:ew {\tl_if_head_is_group_p:n{#3}} #3 \q_recursion_stop
}
\cs_generate_variant:Nn\__erw_last_q:nnw{e}
```

4 predicate

```
\msg_new:nnn{__erw}{predicate-empty}
{empty~expression~in~predicate}
\prg_new_conditional:Npnn
\erw_and_tl:nn
#1 % <predicate expression>
#2 % <tokens>
{p}
{%^^A
  \__erw_and_tl:nw {#1}#2 \q_recursion_tail\q_recursion_stop
}
\cs_new:Npn
\__erw_and_tl:nw
#1 % <predicate expression>
#2 % <value>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#2}
  {  \prg_return_true: }
  \__erw_and_tl:nnw
  {#1} % <predicate expression>
  #2 % <value>
  \q_recursion_stop
}
\cs_new:Npn
\__erw_and_tl:nnw
#1 % <predicate expression>
#2 % <value>
#3 % <rest>
\q_recursion_stop
{%
  \bool_if:nTF
  {#1{#2}}
  {\__erw_and_tl:nw{#1}#3\q_recursion_stop}
  { \prg_return_false: }
}
\cs_new:Npn \__erw_new_compare_p:nnn
#1 % <name>
```

```
#2 % <signature>
#3 % <code>
{%
  \prg_new_conditional:cnn{#1:#2}
  {p}
  {%
    \bool_if:nTF
    {#3}
    {\prg_return_true:}
    {\prg_return_false:}
  }
}
\keys_define:nn{ __erw }
{
  new_compare_p.code:n = {\__erw_new_compare_p:nnn#1}
}
\erw_keys_set:n
{%
  new_compare_p =
  {erw_compare} % <name>
  {nNnNn}
  { \__erw_compare:eecN{ #2{#3} }{ #2{#5} }{ #1:nNn }#4 }
}
\cs_new:Npn
\__erw_compare:nnNN
#1 % <first>
#2 % <second>
#3 % <predicate>
#4 % <operator>
{ #3{ #1 }#4{ #2 } }
\cs_generate_variant:Nn\__erw_compare:nnNN{eec}
\erw_keys_set:n
{%
  new_compare_p =
  {erw_int_incr}
  {nn}
  {\exp_args:Ne
    \int_compare_p:nNn{ \int_eval:n{#1+1} } = {#2} }
}
```

5 keyval

```
\cs_new:Npn\__erw_keyval_key:w #1 = #2 \q_recursion_stop{#1}
\cs_new:Npn\__erw_keyval_value:w #1 = #2 \q_recursion_stop{#2}
\cs_new:Npn \erw_keyval_key:n#1{\__erw_keyval_key:w #1 \q_recursion_stop}
\cs_new:Npn \erw_keyval_value:n#1{\__erw_keyval_value:w #1 \q_recursion_stop}
\cs_new:Npn \erw_keyval:nn#1#2{ #1 = #2 }
\erw_keys_set:n
{
  new_compare_p = {erw_key_compare}
  {nNn}{ \erw_compare_p:nNnNn
    {int_compare_p}\erw_keyval_key:n{#1}#2{#3} },
  new_compare_p = {erw_key_compare}
  {n}{ \erw_compare_recurse_p:nnNN{#1}
```

```
    {int_compare_p}\erw_keyval_key:n< }
}
```

6 op's on list

```
\cs_new:Npn
\erw_remove_first:n
#1 % <tokenlist>
{\erw_remove_first_q:w#1\q_recursion_tail\q_recursion_stop}
\cs_generate_variant:Nn\erw_remove_first:n{e}
\cs_new:Npn
\erw_remove_last:n
#1 % <tokenlist>
{\erw_remove_last_q:w#1\q_recursion_tail\q_recursion_stop}
\cs_generate_variant:Nn\erw_remove_last:n{e}
\cs_new:Npn
\erw_first:n
#1
{\erw_first_q:w#1\q_recursion_tail\q_recursion_stop}
\cs_generate_variant:Nn\erw_first:n{e}
\cs_new:Npn
\erw_last:n
#1 % <tokenlist>
{\erw_last_q:w#1\q_recursion_tail\q_recursion_stop}
\cs_generate_variant:Nn\erw_last:n{e}
\cs_new:Npn
\erw_adjacent_insert:nn
#1 % <list>
#2 % <separator>
{%
  \erw_first:n{#1}
  \erw_swap:en
  { \erw_remove_first:n{#1} }
  {%
    \__erw_adjacent_insert:nw
    {#2} % <separator>
  }
  \q_recursion_tail
  \q_recursion_stop
}
\cs_generate_variant:Nn\erw_adjacent_insert:nn{e}
\cs_new:Npn
\__erw_adjacent_insert:nw
#1 % <separator>
#2 % <rest>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#2}
  \__erw_adjacent_insert:new {#1}{\tl_if_head_is_group_p:n{#2}}#2 \q_recursion_stop
}
\cs_new:Npn
\__erw_adjacent_insert:nnw
#1 % <separator>
#2 % <head is group>
```

```
#3 % <head>
#4 % <rest>
\q_recursion_stop
{%
  #1\bool_if:nTF{#2}{{#3}}{#3}
  \__erw_adjacent_insert:nw{#1}#4\q_recursion_stop
}
\cs_generate_variant:Nn\__erw_adjacent_insert:nnw{ne}

\cs_new:Npn
\erw_clist_tl:nn
#1 % <bool>
#2 % <list>
{ \erw_clist_tl:nnw {#1} #2 \q_recursion_tail\q_recursion_stop }
\cs_new:Npn
\erw_clist_tl:nnw #1 #2\q_recursion_stop
{\quark_if_recursion_tail_stop:n{#2}
  \erw_clist_tl:nenw {#1}
  {\tl_if_head_is_group_p:n{#2}} #2 \q_recursion_stop}
\cs_generate_variant:Nn\erw_clist_tl:nnw{ne}
\cs_new:Npn
\erw_clist_tl:nnnw
#1 % <bool>
#2 % <head is group>
#3 % <head>
#4 % <rest>
\q_recursion_stop
{
  \quark_if_recursion_tail_stop_do:nn{#4}
  {%
    \bool_if:nTF
    {\bool_lazy_and_p:nn{#1}{#2}}
    {{#3}}{#3}
  }
  \bool_if:nTF{\bool_lazy_and_p:nn{#1}{#2}}
  {{#3}}{#3},
  \erw_clist_tl:nnw {#1} #4 \q_recursion_stop
}
\cs_generate_variant:Nn\erw_clist_tl:nnnw{ne}
\prg_new_conditional:Npnn
\erw_if_in_clist:nn
#1 % <value>
#2 % <clist>
{p}
{ \__erw_clist_if_in:nw {#1} #2, \q_recursion_tail \q_recursion_stop }
\cs_new:Npn
\__erw_clist_if_in:nw #1 #2 \q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#2}
  \__erw_clist_if_in:nnw {#1} #2 \q_recursion_stop
}
\cs_new:Nn
\__erw_clist_if_in:nn
{\__erw_clist_if_in:nw{#1} #2 \q_recursion_stop}
\cs_new:Npn
```

```
\__erw_clist_if_in:nnw #1 #2, #3 \q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#3}
  {%
    \str_if_eq:nnTF{#1}{#2}
    {\prg_return_true:}{\prg_return_false:}
  }
  \str_if_eq:nnTF{#1}{#2}
  {\prg_return_true:}
  {\__erw_clist_if_in:nw {#1} #3 \q_recursion_stop}
  \__erw_empty:w\q_recursion_stop
}
```

7 algo

7.1 split

```
\cs_new:Npn
\erw_split_even:n
#1 % <tokenlist>
{%
  \tl_if_empty:nF{#1}
  {%
    \exp_last_unbraced:Ne
    \__erw_split_even:nnnw
    {%
      {\__erw_split_even_threshold:n{#1}} % <count>
      {\tl_if_head_is_group_p:n{#1}} % <head is group>
    }
    #1 % <tokenlist>
    \q_recursion_tail
    \q_recursion_stop
  }
}
\cs_generate_variant:Nn\erw_split_even:n{e}
\cs_new:Npn
\__erw_split_even_threshold:n
#1 % <tokenlist>
{\exp_args:Ne
  \int_div_round:nn{\tl_count:n{#1}}{2}}
\cs_new:Npn
\__erw_split_even:nnnw
#1 % <threshold>
#2 % <head is group>
#3 % <head>
#4 % <rest>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#4}
  { { \bool_if:nTF{#2}{{#3}}{#3} }{} }
  \exp_last_unbraced:Ne
  \__erw_split_even:nnnnw
  {%
    {1} % <left size>
    { \tl_if_head_is_group_p:n{#4} }
    {#1} % <threshold count>
    { \bool_if:nTF{#2}{{#3}}{#3} } % <left list>
  }
  #4 % <right list>
  \q_recursion_stop
}
\cs_new:Npn
\__erw_split_even:nnnnw
#1 % <left size>
#2 % <right head is group>
#3 % <threshold count>
#4 % <left list>
#5 % <right head>
#6 % <right rest>
\q_recursion_stop
{%
  \bool_if:nTF
  { \int_compare_p:nNn {#1}<{#3} }
  {%
    \exp_last_unbraced:Ne
    \__erw_split_even:nnnnw
    {
      { \int_eval:n{#1+1} } % <left size>
      { \tl_if_head_is_group_p:n{#6} } % <right head is group>
      {#3} % <threshold count>
      {#4\bool_if:nTF{#2}{{#5}}{#5}} % <left list>
    }
    #6
    \q_recursion_stop
  }
  {%
    {#4}
    {%
      \bool_if:nTF{#2}{{#5}}{#5}
      \erw_remove_last_q:w#6\q_recursion_stop\erw_last_q:w#6\q_recursion_stop}
  }
}
```

7.2 thread sort

```
\cs_new:Npn
\erw_thread_sort:nnNn
#1 % <first sorted list>
#2 % <second sorted list>
#3 % <compare predicate name>
#4 % <compare operator>
{%
  \__erw_thread_sort:nNnnn
  {#3} % <compare predicate name>
  #4 % <compare operator>
  {\c_empty_tl} % <accum>
  {#1}
  {#2}
}
```

```
\cs_generate_variant:Nn\erw_thread_sort:nnNn{ee}
\cs_new:Npn
\__erw_thread_sort:nNnnn
#1 % <compare predicate name>
#2 % <compare operator>
#3 % <sorted>
#4 % <first>
#5 % <second>
{%
  \__erw_thread_sort:nNnww
  {#1} % <compare predicate name>
  {#2} % <compare operator>
  {#3} % <sorted>
  #4 \q_recursion_tail% <first>
  \q_stop
  #5 \q_recursion_tail% <second>
  \q_recursion_stop
}
\cs_generate_variant:Nn\__erw_thread_sort:nNnnn{nNeee}
\cs_new:Npn
\__erw_thread_sort:nNnww
#1 % <compare predicate name>
#2 % <compare operator>
#3 % <sorted>
#4 % <first>
\q_stop
#5 % <second>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#4}
  { #3 \erw_all_q:w #5 \q_recursion_stop }
  \quark_if_recursion_tail_stop_do:nn{#5}
  { #3 \erw_all_q:w #4 \q_recursion_stop }
  \__erw_thread_sort:nNneeww
  {#1}#2{#3}
  { \tl_if_head_is_group_p:n{#4} }
  { \tl_if_head_is_group_p:n{#5} }
  #4\q_stop
  #5\q_recursion_stop
}
\cs_new:Npn
\__erw_thread_sort:nNnnnww
#1 % <compare predicate name>
#2 % <compare operator>
#3 % <sorted>
#4 % <head is begin>
#5 % <head is begin>
#6 % <first head>
#7 % <first rest>
\q_stop
#8 % <second head>
#9 % <second rest>
\q_recursion_stop
{%
```

```
  \bool_if:nTF
  { \use:c{#1:nNn}{#6}#2{#8} }
  {%
    \__erw_thread_sort:nNeee
    {#1}
    #2
    {#3\bool_if:nTF{#4}{{#6}}{#6}}
    {\erw_all_q:w#7\q_recursion_stop}
    {\bool_if:nTF{#5}{{#8}}{#8}\erw_all_q:w#9\q_recursion_stop}
  }
  {%
    \__erw_thread_sort:nNeee
    {#1}
    #2
    {#3\bool_if:nTF{#5}{{#8}}{#8}}
    {\bool_if:nTF{#4}{{#6}}{#6}\erw_all_q:w#7\q_recursion_stop}
    {\erw_all_q:w#9\q_recursion_stop}
  }
}
\cs_generate_variant:Nn\__erw_thread_sort:nNnnnww{nNnee}
```

7.3 merge sort

```
\cs_new:Npn
\erw_merge_sort:nNn
#1 % <compare predicate name>
#2 % <compare operator>
#3 % <unsorted list>
{%
  \tl_if_empty:nF{#3}
  {%
    \__erw_sort_merge:enNw
    {\tl_if_head_is_group_p:n{#3}} % <head is group>
    {#1} % <compare predicate name>
    #2 % <compare operator>
    #3 % <unsorted list>
    \q_recursion_tail
    \q_recursion_stop
  }
}
\cs_generate_variant:Nn\erw_merge_sort:nNn{nNe}
\cs_new:Npn
\__erw_sort_merge:nnNw
#1 % <head is group>
#2 % <compare predicate name>
#3 % <compare operator>
#4 % <unsorted list head>
#5 % <unsorted list rest>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#5}
  { \bool_if:nTF{#1}{{#4}}{#4} }
  \exp_last_unbraced:Ne
  \__erw_sort_merge:nnnN
  {%
  \erw_split_even:e
  {%
   \bool_if:nTF{#1}{{#4}}{#4}
   \erw_all_q:w#5\q_recursion_stop
  }
 } % {<first sorted list>}{<second sorted list>}
 {#2} % <compare predicate name>
 #3 % <compare operator>
 \__erw_empty:w \q_recursion_stop
}
\cs_generate_variant:Nn\__erw_sort_merge:nnNw{e}
\cs_new:Npn
\__erw_sort_merge:nnnN
#1 % <left unsorted list>
#2 % <right unsorted list>
#3 % <compare predicate name>
#4 % <compare operator>
{%
 \erw_thread_sort:eeNn
 {%
  \__erw_sort_merge:enNw
  {\tl_if_head_is_group_p:n{#1}}
  {#3} % <compare predicate name>
  #4 % <compare operator>
  #1 % <unsorted list>
  \q_recursion_tail
  \q_recursion_stop
 } % <first sorted list>
 {%
  \__erw_sort_merge:enNw
  {\tl_if_head_is_group_p:n{#2}}
  {#3} % <compare predicate name>
  #4 % <compare operator>
  #2 % <unsorted list>
  \q_recursion_tail
  \q_recursion_stop
 } % <second sorted list>
 {#3} % <compare predicate name>
 #4 % <operator>
}
```

7.4 filter

```
\msg_new:nnn{__erw}{tokenlist-incr}
{expecting~an~ascending~tokenlist~got~#1~followed~by~#2}
\cs_new:Npn
\__erw_filter_uniq:nnw
#1 % <compare predicate>
#2 % <greatest>
#3 % <tokenlist>
\q_recursion_stop
{ %
 \quark_if_recursion_tail_stop:n{#3}
 \__erw_filter_uniq_aux:nnw{#1}{#2}#3\q_recursion_stop}
\cs_new:Npn
```

```
\__erw_filter_uniq_aux:nw
#1 % <compare predicate>
#2 % <tokenlist head>
#3 % <tokenlist rest>
\q_recursion_stop
{%
  {#2}
  \__erw_filter_uniq:nnw
  {#1} % <compare predicate>
  {#2} #3 % <tokenlist>
  \q_recursion_stop }
\cs_new:Npn
\__erw_filter_uniq_aux:nnw
#1 % <compare predicate>
#2 % <last>
#3 % <head token>
#4 % <rest token>
\q_recursion_stop
{ %
  \bool_if:nTF{\use:c{#1:nNn}{#3}<{#2}}
  {\msg_error:nnnn{__erw}{tokenlist-incr}{#2}{#3}}
  {%
    \bool_if:nF
    {\use:c{#1:nNn}{#3}={#2}}
% ^^A    {{#3}}
{\tl_if_single_token:nTF{#3}{#3}{{#3}}}
}
\quark_if_recursion_tail_stop:n{#4}
% ^^A  \__erw_filter_uniq:nnw{#1}{#3}#4\q_recursion_stop }
\__erw_filter_uniq:nnw{#1}{#3}#4\q_recursion_stop }
\cs_new:Npn
\__erw_filter_uniq:nw
#1 % <compare predicate>
#2 % <tokenlist>
{%
  \quark_if_recursion_tail_stop_do:nn{#2}{\c_empty_tl}
  \__erw_filter_uniq_aux:nw {#1}#2 \q_recursion_stop}
\cs_new:Npn
\erw_filter_uniq:nn
#1 % <compare predicate>
#2 % <tokenlist>
{%
  \__erw_filter_uniq_aux:nw
  {#1} % <compare predicate>
  #2
  \q_recursion_tail % <head token>
  \q_recursion_stop}
\cs_new:Npn
\erw_filter_uniq:n
#1 % <ascending integers>
{ \erw_filter_uniq:nn{int_compare_p}{#1} }
\cs_generate_variant:Nn\erw_filter_uniq:nn{ne}
```

8 code

```
\keys_define:nn{__erw}
{ clist_map_inline.code:n = \__erw_map_inline_clist:nnn#1 }
\cs_new_protected:Npn
\__erw_map_inline_clist:nnn
#1 % <clist>
#2 % <signature>
#3 % <code>
{
  \cs_new_protected:cn
  {__erw_do:#2}{#3}
  \clist_map_inline:nn
  {#1}
  {\use:c{__erw_do:#2}##1}
}
\cs_new:Npn
\erw_parameter:n
#1 %^^A  <arity>
{## #1}
\cs_new:Npn
\__erw_parameter_aux:nn
#1 % <finish>
#2 % <start>
{ \int_step_function:nnN {#2}{#1}\erw_parameter:n}
\cs_new:Npn
\erw_parameter:nn
#1 % <start>
#2 % <count>
{%
  \exp_args:Ne
  \__erw_parameter_aux:nn
  {\int_eval:n{#1+#2-1}}{#1}}
\cs_new:Npn
\erw_argument:nn
#1 % <position>
#2 % <signature>
{\__erw_argument:nw{#1}#2\q_recursion_tail\q_recursion_stop}
\cs_new:Npn
\__erw_argument_unit:nn
#1 % <position>
#2 % <n|N>
{\use:c{__erw_argument_#2:w} #1 \q_recursion_stop}
\cs_new:Npn\__erw_argument_n:w #1 \q_recursion_stop{{## #1}}
\cs_new:Npn\__erw_argument_N:w #1 \q_recursion_stop{## #1}
\cs_new:Npn
\__erw_argument:nw
#1 % <position>
#2 % <signature list>
\q_recursion_stop
{ \quark_if_recursion_tail_stop:n{#2}
  \__erw_argument:nnw{#1}#2\q_recursion_stop }
\cs_new:Npn
\__erw_argument:nnw
```

```
#1 % <position>
#2 % <n|N>
#3 % <signature rest>
\q_recursion_stop
{%
  \__erw_argument_unit:nn{#1}{#2}
  \exp_args:Ne
  \__erw_argument:nw
  {\erw_int_incr:n{#1}}#3\q_recursion_stop }
```

```
\ProcessKeysOptions{__erw}
\ExplSyntaxOff
⟨/package⟩
```