CVector

0.2.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 cvector Struct Reference

`#include <cvector_interface.h>`

Data Fields

- index_t _size
- index_t _space
- value_t * _vector

3.1.1 Detailed Description

Main struct holding the cvector structure.

Definition at line 190 of file cvector_interface.h.

3.1.2 Field Documentation

3.1.2.1 _size

`index_t _size`

Definition at line 191 of file cvector_interface.h.

3.1.2.2 _space

```
index_t _space
```

Definition at line 192 of file cvector_interface.h.

3.1.2.3 _vector

```
value_t* _vector
```

Definition at line 193 of file cvector_interface.h.

The documentation for this struct was generated from the following file:

- lib/cvector_interface.h

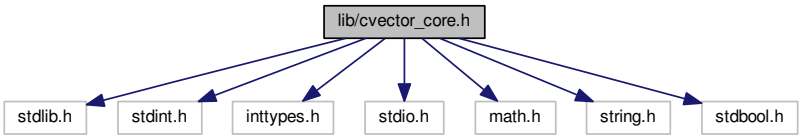
Chapter 4

File Documentation

4.1 lib/cvector_core.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <inttypes.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdbool.h>
```

Include dependency graph for cvector_core.h:



Functions

- void __cvector_setspace (cvector *p_cvector, index_t new_space)
- void __cvector_shrink (cvector *p_cvector)
- void __cvector_extend (cvector *p_cvector)
- void cvector_readjust (cvector *p_cvector)
- void cvector_addspace (cvector *p_cvector)
- cvector * cvector_new ()
- cvector * cvector_new_space (index_t space)
- cvector * cvector_new_copy (cvector *p_original)
- cvector * cvector_new_copy_space (cvector *p_original, index_t space)
- void cvector_free (cvector *p_cvector)
- void cvector_free_func (cvector *p_vector, void(*free_value)(value_t))
- index_t cvector_getsize (cvector *p_cvector)
- void cvector_add (cvector *p_cvector, value_t value)
- void cvector_addi (cvector *p_cvector, value_t value, index_t index)

- void cvector_insert (cvector *p_cvector, value_t value)
- value_t cvector_remove (cvector *p_cvector)
- value_t cvector_removei (cvector *p_cvector, index_t index)
- value_t cvector_drop (cvector *p_cvector)
- void cvector_clear (cvector *p_cvector)
- value_t cvector_get (cvector *p_cvector, index_t index)
- value_t cvector_safeget (cvector *p_cvector, index_t index)
- void cvector_set (cvector *p_cvector, value_t value, index_t index)
- void cvector_safeset (cvector *p_cvector, value_t value, index_t index)
- void cvector_appendto (cvector *p_cvector, cvector *p_add)
- cvector * cvector_concat (cvector *p_cvector_1, cvector *p_cvector_2)
- cvector * cvector_reversed (cvector *p_cvector)
- hash_t cvector_hash (cvector *p_cvector, hash_t(*hash_value)(value_t))
- bool cvector_equal (cvector *p_cvector_1, cvector *p_cvector_2)
- bool cvector_equal_func (cvector *p_cvector_1, cvector *p_cvector_2, bool(*equal_value)(value_t, value_t))
- value_t * cvector_toarray (cvector *p_cvector)
- bool cvector_replace (cvector *p_cvector, value_t original, value_t replacement)
- bool cvector_replace_func (cvector *p_cvector, value_t original, value_t replacement, bool(*equal_↩ value)(value_t, value_t))
- void cvector_sort (cvector *p_cvector, int(*comp_value)(const void *, const void *))
- index_t cvector_indexof (cvector *p_cvector, value_t value)
- index_t cvector_indexof_func (cvector *p_cvector, value_t value, bool(*equal_value)(value_t, value_t))
- bool cvector_in (cvector *p_cvector, value_t value)
- bool cvector_in_func (cvector *p_cvector, value_t value, bool(*equal_value)(value_t, value_t))
- cvector * cvector_slice (cvector *p_cvector, index_t from, index_t to, index_t step)
- value_t * cvector_slicetoarray (cvector *p_cvector, index_t from, index_t to, index_t step)

4.1.1 Function Documentation

4.1.1.1 __cvector_extend()

```
void __cvector_extend (
            cvector * p_cvector )
```

Extends the specified cvector without any check about its size.

Parameters

p_cvector	a pointer to the cvector

Definition at line 51 of file cvector_core.h.

4.1.1.2 __cvector_setspace()

```
void __cvector_setspace (
            cvector * p_cvector,
            index_t new_space )
```

Sets space of the specified cvector to new_space

Parameters

p_cvector	a pointer to the cvector
new_space	the new space for the specified cvector

Definition at line 30 of file cvector_core.h.

4.1.1.3 __cvector_shrink()

```
void __cvector_shrink (
            cvector * p_cvector )
```

Shrinks the specified cvector without any check about its size.

Parameters

p_cvector	a pointer to the cvector

Definition at line 42 of file cvector_core.h.

4.1.1.4 cvector_add()

```
void cvector_add (
            cvector * p_cvector,
            value_t value )
```

Adds the specified element at the end of the cvector.

Parameters

p_cvector	a pointer to the cvector
value	the value to push at the end of the cvector

Definition at line 202 of file cvector_core.h.

4.1.1.5 cvector_addi()

```
void cvector_addi (
            cvector * p_cvector,
```

```
value_t value,
index_t index )
```

Adds the specified element a the position index in the cvector, and shift following elements to the right.

Parameters

p_cvector	a pointer to the cvector
value	the value to push a the position index in the cvector
index	the index where the specified value will be inserted

Definition at line 219 of file cvector_core.h.

4.1.1.6 cvector_addspace()

```
void cvector_addspace (
            cvector * p_cvector )
```

Adds space (according to the DEFAULT_ADDSPACE_FACTOR) to the specified cvector.

Parameters

p_cvector	a pointer to the cvector to extend.

Definition at line 75 of file cvector_core.h.

4.1.1.7 cvector_appendto()

```
void cvector_appendto (
            cvector * p_cvector,
            cvector * p_add )
```

Appends element of the cvector pointed by p_add at the end of the cvector pointed by p_cvector.

Parameters

p_cvector	a pointer to the cvector where elements will be appended
p_add	a pointer to the cvector containing elements to copy

Definition at line 440 of file cvector_core.h.

4.1.1.8 cvector_clear()

```
void cvector_clear (
            cvector * p_cvector )
```

Removes all elements of the cvector without changing its space (that is to say without calling cvector_readjust).

Parameters

p_cvector	a pointer to the cvector

Definition at line 330 of file cvector_core.h.

4.1.1.9 cvector_concat()

```
cvector* cvector_concat (
            cvector * p_cvector_1,
            cvector * p_cvector_2 )
```

Returns a new cvector which is the concatenation of the two specified cvectors

Parameters

p_cvector_1	a pointer to the first cvector to concatenate
p_cvector_2	a pointer to the first cvector to concatenate

Returns

a pointer to the resulting cvector

Definition at line 461 of file cvector_core.h.

4.1.1.10 cvector_drop()

```
value_t cvector_drop (
            cvector * p_cvector )
```

Removes the first element of the cvector. If the cvector is empty, prints an error anrd returns DEFAULT_VALUE.

Parameters

p_cvector	a pointer to the cvector

Returns

the remove (first) element, or DEFAULT_VALUE if an error occurs

Definition at line 321 of file cvector_core.h.

4.1.1.11 cvector_equal()

```
bool cvector_equal (
            cvector * p_cvector_1,
            cvector * p_cvector_2 )
```

Returns true iif both specified cvectors are equal.

Parameters

p_cvector_1	a pointer to the first cvector to test
p_cvector_2	a pointer to the second cvector to test

Returns

true if both specified cvectors are equal, false otherwise

Definition at line 517 of file cvector_core.h.

4.1.1.12 cvector_equal_func()

```
bool cvector_equal_func (
            cvector * p_cvector_1,
            cvector * p_cvector_2,
            bool(*)(value_t, value_t) equal_value )
```

Returns true iif both specified cvectors are equal according to the specified test function for values.

Parameters

p_cvector_1	a pointer to the first cvector to test
p_cvector_2	a pointer to the second cvector to test
equal_value	the test function for values. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

true if both specified cvectors are equal according to the test function, false otherwise

Definition at line 539 of file cvector_core.h.

4.1.1.13 cvector_free()

```
void cvector_free (
            cvector * p_cvector )
```

Frees the specified cvector.

Parameters

p_cvector	a pointer to the cvector to free

Definition at line 169 of file cvector_core.h.

4.1.1.14 cvector_free_func()

```
void cvector_free_func (
            cvector * p_vector,
            void(*)(value_t) free_value )
```

Applies the specified free function of each value of the cvector, and then frees it too.

Parameters

p_vector	a pointer to the cvector to free
free_value	the function to free each value of the cvector

Definition at line 180 of file cvector_core.h.

4.1.1.15 cvector_get()

```
value_t cvector_get (
            cvector * p_cvector,
            index_t index )
```

Returns the value at the specified index in the cvector. Prints an error message and returns DEFAULT_VALUE if the specified index is invalid.

Parameters

p_cvector	a pointer to the cvector
index	the index of the value to get

Returns

the desired value if the index is correct, DEFAULT_VALUE otherwise

Definition at line 341 of file cvector_core.h.

4.1.1.16 cvector_getsize()

```
index_t cvector_getsize (
            cvector * p_cvector )
```

Size getter. Returns the size of the cvector.

Parameters

p_cvector	a pointer to the cvector

Returns

the size of the cvector

Definition at line 193 of file cvector_core.h.

4.1.1.17 cvector_hash()

```
hash_t cvector_hash (
            cvector * p_cvector,
            hash_t(*)(value_t) hash_value )
```

Returns the hash of the specified cvector, using djb2 algorithm by Dan Bernstein, according to the specified hash function for values of the cvector.

Parameters

p_cvector	a pointer to the cvector to hash
hash_value	hash function for values of the cvector. Signature of the hash value function must be hash_t hash_value(value_t value)

Returns

the computed hash of the specified cvector

Definition at line 503 of file cvector_core.h.

4.1.1.18 cvector_in()

```
bool cvector_in (
            cvector * p_cvector,
            value_t value )
```

Returns true iif the specified value was found in the cvector.

Parameters

p_cvector	a pointer to the cvector
value	the value to found

Returns

true if the value was found, false otherwise

Definition at line 676 of file cvector_core.h.

4.1.1.19 cvector_in_func()

```
bool cvector_in_func (
            cvector * p_cvector,
            value_t value,
            bool(*)(value_t, value_t) equal_value )
```

Returns true iif the specified value was found in the cvector according to the specified test function.

Parameters

p_cvector	a pointer to the cvector
value	the value to found
equal_value	the test function to check equality between values. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

true if the value was found, false otherwise

Definition at line 690 of file cvector_core.h.

4.1.1.20 cvector_indexof()

```
index_t cvector_indexof (
            cvector * p_cvector,
            value_t value )
```

Returns the first index where the specified value is found in the cvector. If the value is not found, returns NOT_F↩ OUND_INDEX value.

Parameters

p_cvector	a pointer to the cvector
value	the value to found

Returns

the first index where the specified value was found, or NOT_FOUND_INDEX if it was not found

Definition at line 639 of file cvector_core.h.

4.1.1.21 cvector_indexof_func()

```
index_t cvector_indexof_func (
            cvector * p_cvector,
            value_t value,
            bool(*)(value_t, value_t) equal_value )
```

Returns the first index where the specified value is found, according to the specified test function. Il the value is not found, returns NOT_FOUND_INDEX value.

Parameters

p_cvector	a pointer to the cvector
value	the value to found
equal_value	the test function to check equality between values. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

the first index where the specified value was found, or NOT_FOUND_INDEX if it was not found

Definition at line 660 of file cvector_core.h.

4.1.1.22 cvector_insert()

```
void cvector_insert (
            cvector * p_cvector,
            value_t value )
```

Adds the specified value at the beginning of the cvector, and shift following elements to the right.

Parameters

p_cvector	a pointer to the cvector
value	the value to add at the beginning of the cvector

Definition at line 253 of file cvector_core.h.

4.1.1.23 cvector_new()

```
cvector* cvector_new ( )
```

Creates a new cvector which can hold at the beginning at least DEFAULT_INIT_SPACE elements.

Returns

a pointer to the new cvector

Definition at line 87 of file cvector_core.h.

4.1.1.24 cvector_new_copy()

```
cvector* cvector_new_copy (
            cvector * p_original )
```

Creates a new cvector which is a copy of the specified one.

Parameters

p_original	a pointer to the cvector to copy

Returns

a pointer to the new (clone) cvector

Definition at line 121 of file cvector_core.h.

4.1.1.25 cvector_new_copy_space()

```
cvector* cvector_new_copy_space (
            cvector * p_original,
            index_t space )
```

Creates a new cvector which is a copy of the specified one and which can hold at least space elements.

Parameters

p_original	a pointer to the cvector to copy
space	desired space for the new (clone) cvector. space must be greater or equal than the size of the original cvector

Returns

a pointer to the new (clone) cvector

Definition at line 142 of file cvector_core.h.

4.1.1.26 cvector_new_space()

```
cvector* cvector_new_space (
            index_t space )
```

Creates a new cvector which can hold at the beginning at least space elements.

Parameters

space	desired space for the new cvector

Returns

a pointer to the new cvector

Definition at line 102 of file cvector_core.h.

4.1.1.27 cvector_readjust()

```
void cvector_readjust (
            cvector * p_cvector )
```

Readjusts space of the specified cvector if needed, according to SHRINK_THRESHOLD and EXTEND_THRES↩ HOLD.

Parameters

p_cvector	a pointer to the cvector

Definition at line 61 of file cvector_core.h.

4.1.1.28 cvector_remove()

```
value_t cvector_remove (
            cvector * p_cvector )
```

Removes the last element of the cvector and returns it. If the cvector is empty, prints an error and returns DEFA↩ ULT_VALUE.

Parameters

p_cvector	a pointer to the cvector

Returns

The last value of the cvector if it is not empty, DEFAULT_VALUE otherwise

Definition at line 264 of file cvector_core.h.

4.1.1.29 cvector_removei()

```
value_t cvector_removei (
            cvector * p_cvector,
            index_t index )
```

Removes the element located at the specified index, and returns it. If the cvector is empty or if the index is incorrect, prints an error and returns DEFAULT_VALUE.

Parameters

p_cvector	a pointer to the cvector
index	the index where the element will be removed

Returns

the removed element or DEFAULT_VALUE if an error occurs

Definition at line 284 of file cvector_core.h.

4.1.1.30 cvector_replace()

```
bool cvector_replace (
            cvector * p_cvector,
            value_t original,
            value_t replacement )
```

Replace specified elements in the cvector and returns true if at least one change was made.

Parameters

p_cvector	a pointer to the cvector
original	original value to replace
replacement	replacement value for original

Returns

true if at least one replacement was made, false otherwise

Definition at line 577 of file cvector_core.h.

4.1.1.31 cvector_replace_func()

```
bool cvector_replace_func (
            cvector * p_cvector,
            value_t original,
            value_t replacement,
            bool(*)(value_t, value_t) equal_value )
```

Replace specified elements in the cvector and returns true if at least one change was made. Test between elements of the cvector and original are made with the specified function.

Parameters

p_cvector	a pointer to the cvector
original	original value to replace
replacement	replacement value for original
equal_value	test function used to compare cvector elements and original. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

true if at least one replacement was made, false otherwise

Definition at line 600 of file cvector_core.h.

4.1.1.32 cvector_reversed()

```
cvector* cvector_reversed (
            cvector * p_cvector )
```

Returns a cvector which contains the same elements as the specified one, but in a reversed order.

Parameters

p_cvector	a pointer to the original cvector

Returns

the resulting cvector, containing elements of the specified cvector in a reverse order

Definition at line 481 of file cvector_core.h.

4.1.1.33 cvector_safeget()

```
value_t cvector_safeget (
            cvector * p_cvector,
            index_t index )
```

Returns the value at the specified index in the cvector. Only prints a warning and returns DEFAULT_VALUE if the specified index is invalid.

Parameters

p_cvector	a pointer to the cvector
index	the index of the value to get

Returns

the desired value if the index is correct, DEFAULT_VALUE otherwise

Definition at line 364 of file cvector_core.h.

4.1.1.34 cvector_safeset()

```
void cvector_safeset (
            cvector * p_cvector,
            value_t value,
            index_t index )
```

Sets the value of the element located at the specified position. Only raises warning if the index is invalid, or extends the cvector to be able to set the value at the specified index.

Parameters

p_cvector	a pointer to the cvector
value	the value which will be inserted at the index position
index	the index where the value will be set

Definition at line 410 of file cvector_core.h.

4.1.1.35 cvector_set()

```
void cvector_set (
            cvector * p_cvector,
```

```
            value_t value,
            index_t index )
```

Sets the value of the element located at the specified index. Raises error if the specified index is invalid.

Parameters

p_cvector	a pointer to the cvector
value	the value which will be placed at the index position
index	the index where the value will be set

Definition at line 387 of file cvector_core.h.

4.1.1.36 cvector_slice()

```
cvector* cvector_slice (
            cvector * p_cvector,
            index_t from,
            index_t to,
            index_t step )
```

Returns the slice [|from:to[| of the specified cvector. Prints an error and return NULL if indexes are incorrect.

Parameters

p_cvector	a pointer to the cvector
from	index of the begin of the slice, included
to	index of the end of the slice, excluded
step	step of the slice

Returns

the corresponding (cvector) slice

Definition at line 707 of file cvector_core.h.

4.1.1.37 cvector_slicetoarray()

```
value_t* cvector_slicetoarray (
            cvector * p_cvector,
            index_t from,
            index_t to,
            index_t step )
```

Returns the slice [|from:to[| of the specified cvector as a c-style array. Prints an error and return NULL if indexes are incorrect.

Parameters

p_cvector	a pointer to the cvector
from	index of the begin of the slice, included
to	index of the end of the slice, excluded
step	step of the slice

Returns

the corresponding (c-style array) slice

Definition at line 755 of file cvector_core.h.

4.1.1.38 cvector_sort()

```
void cvector_sort (
            cvector * p_cvector,
            int(*)(const void *, const void *) comp_value )
```

Sorts the elements in the cvector according to the specified comparison function.

Parameters

p_cvector	a pointer to the cvector
comp_value	a comparison function which must have the signature int comp_value(const void ∗p_a, const void ∗p_b) and which must • return -1 if element a should be placed before element b • return 0 if element a and b could be placed at the same position • return 1 if element a should be placed after element b

Definition at line 624 of file cvector_core.h.

4.1.1.39 cvector_toarray()

```
value_t* cvector_toarray (
            cvector * p_cvector )
```

Returns a pointer to a c-style array holding the same elements as the specified cvector.

Parameters

p_cvector	a pointer to the cvector

Returns

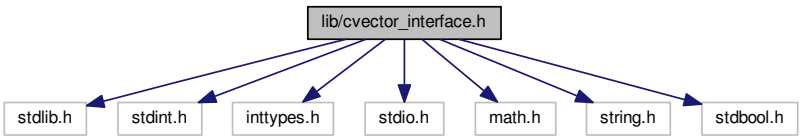
a c-style malloc-ed array holding the same elements as the specified cvector, which must be freed after use

Definition at line 560 of file cvector_core.h.

4.2 lib/cvector_interface.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <inttypes.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdbool.h>
```

Include dependency graph for cvector_interface.h:



Data Structures

- struct cvector

Macros

- #define _CONCAT(a, b) a ## b
- #define CONCAT(a, b) _CONCAT(a, b)
- #define index_t long
- #define NOT_FOUND_INDEX ((index_t) (-1))
- #define ROUND_INDEX(x) ((index_t) (lrint(x)))
- #define CVECTOR_INIT_SPACE 8
- #define CVECTOR_INIT_FACTOR 1.25
- #define CVECTOR_ADDSPACE_FACTOR 2.0
- #define CVECTOR_SHRINK_THRESHOLD 0.5
- #define CVECTOR_SHRINK_FACTOR 0.5
- #define CVECTOR_EXTEND_THRESHOLD 0.90
- #define CVECTOR_EXTEND_FACTOR 2.0
- #define CVECTOR_ERROR(lvl, msg) NULL
- #define CVECTOR_T int
- #define CVECTOR_DEFAULT_VALUE 0
- #define CVECTOR_HASH_T size_t
- #define value_t CVECTOR_T
- #define hash_t CVECTOR_HASH_T

- #define cvector CONCAT(CVECTOR_T, _vect)
- #define __cvector_setspace CONCAT(CVECTOR_T, _vect__setspace)
- #define __cvector_shrink CONCAT(CVECTOR_T, _vect__shrink)
- #define __cvector_extend CONCAT(CVECTOR_T, _vect__extend)
- #define cvector_readjust CONCAT(CVECTOR_T, _vect__readjust)
- #define cvector_addspace CONCAT(CVECTOR_T, _vect__addspace)
- #define cvector_new CONCAT(CVECTOR_T, _vect__new)
- #define cvector_new_space CONCAT(CVECTOR_T, _vect__new_space)
- #define cvector_new_copy CONCAT(CVECTOR_T, _vect__new_copy)
- #define cvector_new_copy_space CONCAT(CVECTOR_T, _vect__new_copy_space)
- #define cvector_free CONCAT(CVECTOR_T, _vect__free)
- #define cvector_getsize CONCAT(CVECTOR_T, _vect__getsize)
- #define cvector_free_func CONCAT(CVECTOR_T, _vect__free_value)
- #define cvector_add CONCAT(CVECTOR_T, _vect__add)
- #define cvector_addi CONCAT(CVECTOR_T, _vect__addi)
- #define cvector_insert CONCAT(CVECTOR_T, _vect__insert)
- #define cvector_remove CONCAT(CVECTOR_T, _vect__remove)
- #define cvector_removei CONCAT(CVECTOR_T, _vect__removei)
- #define cvector_drop CONCAT(CVECTOR_T, _vect__drop)
- #define cvector_clear CONCAT(CVECTOR_T, _vect__clear)
- #define cvector_get CONCAT(CVECTOR_T, _vect__get)
- #define cvector_safeget CONCAT(CVECTOR_T, _vect__safeget)
- #define cvector_set CONCAT(CVECTOR_T, _vect__set)
- #define cvector_safeset CONCAT(CVECTOR_T, _vect__safeset)
- #define cvector_appendto CONCAT(CVECTOR_T, _vect__appendto)
- #define cvector_concat CONCAT(CVECTOR_T, _vect__concat)
- #define cvector_reversed CONCAT(CVECTOR_T, _vect__reversed)
- #define cvector_hash CONCAT(CVECTOR_T, _vect__hash)
- #define cvector_equal CONCAT(CVECTOR_T, _vect__equal)
- #define cvector_equal_func CONCAT(CVECTOR_T, _vect__equal_func)
- #define cvector_toarray CONCAT(CVECTOR_T, _vect__toarray)
- #define cvector_replace CONCAT(CVECTOR_T, _vect__replace)
- #define cvector_replace_func CONCAT(CVECTOR_T, _vect__replace_func)
- #define cvector_sort CONCAT(CVECTOR_T, _vect__sort)
- #define cvector_indexof CONCAT(CVECTOR_T, _vect__indexof)
- #define cvector_indexof_func CONCAT(CVECTOR_T, _vect__indexof_func)
- #define cvector_in CONCAT(CVECTOR_T, _vect__in)
- #define cvector_in_func CONCAT(CVECTOR_T, _vect__in_func)
- #define cvector_slice CONCAT(CVECTOR_T, _vect__slice)
- #define cvector_slicetoarray CONCAT(CVECTOR_T, _vect__slicetoarray)

Typedefs

- typedef struct cvector cvector

Functions

- void __cvector_setspace (cvector *p_cvector, index_t new_space)
- void __cvector_shrink (cvector *p_cvector)
- void __cvector_extend (cvector *p_cvector)
- void cvector_readjust (cvector *p_cvector)
- void cvector_addspace (cvector *p_cvector)
- cvector * cvector_new ()
- cvector * cvector_new_space (index_t space)
- cvector * cvector_new_copy (cvector *p_original)
- cvector * cvector_new_copy_space (cvector *p_original, index_t space)
- void cvector_free (cvector *p_cvector)
- void cvector_free_func (cvector *p_vector, void(*free_value)(value_t))
- index_t cvector_getsize (cvector *p_cvector)
- void cvector_add (cvector *p_cvector, value_t value)
- void cvector_addi (cvector *p_cvector, value_t value, index_t index)
- void cvector_insert (cvector *p_cvector, value_t value)
- value_t cvector_remove (cvector *p_cvector)
- value_t cvector_removei (cvector *p_cvector, index_t index)
- value_t cvector_drop (cvector *p_cvector)
- void cvector_clear (cvector *p_cvector)
- value_t cvector_get (cvector *p_cvector, index_t index)
- value_t cvector_safeget (cvector *p_cvector, index_t index)
- void cvector_set (cvector *p_cvector, value_t value, index_t index)
- void cvector_safeset (cvector *p_cvector, value_t value, index_t index)
- void cvector_appendto (cvector *p_cvector, cvector *p_add)
- cvector * cvector_concat (cvector *p_cvector_1, cvector *p_cvector_2)
- cvector * cvector_reversed (cvector *p_cvector)
- hash_t cvector_hash (cvector *p_cvector, hash_t(*hash_value)(value_t))
- bool cvector_equal (cvector *p_cvector_1, cvector *p_cvector_2)
- bool cvector_equal_func (cvector *p_cvector_1, cvector *p_cvector_2, bool(*equal_value)(value_t, value_t))
- value_t * cvector_toarray (cvector *p_cvector)
- bool cvector_replace (cvector *p_cvector, value_t original, value_t replacement)
- bool cvector_replace_func (cvector *p_cvector, value_t original, value_t replacement, bool(*equal_↩ value)(value_t, value_t))
- void cvector_sort (cvector *p_cvector, int(*comp_value)(const void *, const void *))
- index_t cvector_indexof (cvector *p_cvector, value_t value)
- index_t cvector_indexof_func (cvector *p_cvector, value_t value, bool(*equal_value)(value_t, value_t))
- bool cvector_in (cvector *p_cvector, value_t value)
- bool cvector_in_func (cvector *p_cvector, value_t value, bool(*equal_value)(value_t, value_t))
- cvector * cvector_slice (cvector *p_cvector, index_t from, index_t to, index_t step)
- value_t * cvector_slicetoarray (cvector *p_cvector, index_t from, index_t to, index_t step)

4.2.1 Macro Definition Documentation

4.2.1.1 __cvector_extend

```
#define __cvector_extend CONCAT(CVECTOR_T, _vect__extend)
```

Definition at line 144 of file cvector_interface.h.

4.2.1.2 __cvector_setspace

```
#define __cvector_setspace CONCAT(CVECTOR_T, _vect__setspace)
```

Definition at line 142 of file cvector_interface.h.

4.2.1.3 __cvector_shrink

```
#define __cvector_shrink CONCAT(CVECTOR_T, _vect__shrink)
```

Definition at line 143 of file cvector_interface.h.

4.2.1.4 _CONCAT

```
#define _CONCAT(
              a,
              b ) a ## b
```

Definition at line 25 of file cvector_interface.h.

4.2.1.5 CONCAT

```
#define CONCAT(
              a,
              b ) _CONCAT(a, b)
```

Definition at line 26 of file cvector_interface.h.

4.2.1.6 cvector

```
#define cvector CONCAT(CVECTOR_T, _vect)
```

Definition at line 141 of file cvector_interface.h.

4.2.1.7 cvector_add

```
#define cvector_add CONCAT(CVECTOR_T, _vect__add)
```

Definition at line 154 of file cvector_interface.h.

4.2.1.8 cvector_addi

```
#define cvector_addi CONCAT(CVECTOR_T, _vect__addi)
```

Definition at line 155 of file cvector_interface.h.

4.2.1.9 cvector_addspace

```
#define cvector_addspace CONCAT(CVECTOR_T, _vect__addspace)
```

Definition at line 146 of file cvector_interface.h.

4.2.1.10 CVECTOR_ADDSPACE_FACTOR

```
#define CVECTOR_ADDSPACE_FACTOR 2.0
```

Space factor used when a cvector becomes too short to hold additional values. It means that the new cvector will have a space for ADDSPACE_FACTOR

- the old space.

Definition at line 63 of file cvector_interface.h.

4.2.1.11 cvector_appendto

```
#define cvector_appendto CONCAT(CVECTOR_T, _vect__appendto)
```

Definition at line 165 of file cvector_interface.h.

4.2.1.12 cvector_clear

```
#define cvector_clear CONCAT(CVECTOR_T, _vect__clear)
```

Definition at line 160 of file cvector_interface.h.

4.2.1.13 cvector_concat

```
#define cvector_concat CONCAT(CVECTOR_T, _vect__concat)
```

Definition at line 166 of file cvector_interface.h.

4.2.1.14 CVECTOR_DEFAULT_VALUE

```
#define CVECTOR_DEFAULT_VALUE 0
```

Default value for the type of this instance of cvector, used when an error occurs and when a function needs to return a value.

Definition at line 132 of file cvector_interface.h.

4.2.1.15 cvector_drop

```
#define cvector_drop CONCAT(CVECTOR_T, _vect__drop)
```

Definition at line 159 of file cvector_interface.h.

4.2.1.16 cvector_equal

```
#define cvector_equal CONCAT(CVECTOR_T, _vect__equal)
```

Definition at line 169 of file cvector_interface.h.

4.2.1.17 cvector_equal_func

```
#define cvector_equal_func CONCAT(CVECTOR_T, _vect__equal_func)
```

Definition at line 170 of file cvector_interface.h.

4.2.1.18 CVECTOR_ERROR

```
#define CVECTOR_ERROR(
            lvl,
            msg ) NULL
```

Print debug function called when some error or log message needs to be printed on the screen or the log. The function signature must be void print_debug(int level, const char ∗message)

Definition at line 110 of file cvector_interface.h.

4.2.1.19 CVECTOR_EXTEND_FACTOR

```
#define CVECTOR_EXTEND_FACTOR 2.0
```

Space factor used when a extend operation is triggered. It means that the new space of the cvector will be EXTE↩ ND_FACTOR ∗ the current space.

Definition at line 99 of file cvector_interface.h.

4.2.1.20 CVECTOR_EXTEND_THRESHOLD

```
#define CVECTOR_EXTEND_THRESHOLD 0.90
```

Threshold from which the cvector will be extended in a readjust operation. It means that if the current size of the cvector is above EXTEND_THRESHOLD

- its space, it will be extended. Set to above 1 to prevent extend during readjust operations.

Definition at line 91 of file cvector_interface.h.

4.2.1.21 cvector_free

```
#define cvector_free CONCAT(CVECTOR_T, _vect__free)
```

Definition at line 151 of file cvector_interface.h.

4.2.1.22 cvector_free_func

```
#define cvector_free_func CONCAT(CVECTOR_T, _vect__free_value)
```

Definition at line 153 of file cvector_interface.h.

4.2.1.23 cvector_get

```
#define cvector_get CONCAT(CVECTOR_T, _vect__get)
```

Definition at line 161 of file cvector_interface.h.

4.2.1.24 cvector_getsize

```
#define cvector_getsize CONCAT(CVECTOR_T, _vect__getsize)
```

Definition at line 152 of file cvector_interface.h.

4.2.1.25 cvector_hash

```
#define cvector_hash CONCAT(CVECTOR_T, _vect__hash)
```

Definition at line 168 of file cvector_interface.h.

4.2.1.26 CVECTOR_HASH_T

```
#define CVECTOR_HASH_T size_t
```

Definition at line 136 of file cvector_interface.h.

4.2.1.27 cvector_in

```
#define cvector_in CONCAT(CVECTOR_T, _vect__in)
```

Definition at line 177 of file cvector_interface.h.

4.2.1.28 cvector_in_func

```
#define cvector_in_func CONCAT(CVECTOR_T, _vect__in_func)
```

Definition at line 178 of file cvector_interface.h.

4.2.1.29 cvector_indexof

```
#define cvector_indexof CONCAT(CVECTOR_T, _vect__indexof)
```

Definition at line 175 of file cvector_interface.h.

4.2.1.30 cvector_indexof_func

```
#define cvector_indexof_func CONCAT(CVECTOR_T, _vect__indexof_func)
```

Definition at line 176 of file cvector_interface.h.

4.2.1.31 CVECTOR_INIT_FACTOR

```
#define CVECTOR_INIT_FACTOR 1.25
```

Space factor used when a copy of cvector is created, or a concatenation of two cvectors. It means that the resulting array will have a space for INIT_FACTOR $*$ actual size items.

Definition at line 54 of file cvector_interface.h.

4.2.1.32 CVECTOR_INIT_SPACE

```
#define CVECTOR_INIT_SPACE 8
```

Space in element units of a fresh created cvector, if no space was specified.

Definition at line 45 of file cvector_interface.h.

4.2.1.33 cvector_insert

```
#define cvector_insert CONCAT(CVECTOR_T, _vect__insert)
```

Definition at line 156 of file cvector_interface.h.

4.2.1.34 cvector_new

```
#define cvector_new CONCAT(CVECTOR_T, _vect__new)
```

Definition at line 147 of file cvector_interface.h.

4.2.1.35 cvector_new_copy

```
#define cvector_new_copy CONCAT(CVECTOR_T, _vect__new_copy)
```

Definition at line 149 of file cvector_interface.h.

4.2.1.36 cvector_new_copy_space

```
#define cvector_new_copy_space CONCAT(CVECTOR_T, _vect__new_copy_space)
```

Definition at line 150 of file cvector_interface.h.

4.2.1.37 cvector_new_space

```
#define cvector_new_space CONCAT(CVECTOR_T, _vect__new_space)
```

Definition at line 148 of file cvector_interface.h.

4.2.1.38 cvector_readjust

```
#define cvector_readjust CONCAT(CVECTOR_T, _vect__readjust)
```

Definition at line 145 of file cvector_interface.h.

4.2.1.39 cvector_remove

```
#define cvector_remove CONCAT(CVECTOR_T, _vect__remove)
```

Definition at line 157 of file cvector_interface.h.

4.2.1.40 cvector_removei

```
#define cvector_removei CONCAT(CVECTOR_T, _vect__removei)
```

Definition at line 158 of file cvector_interface.h.

4.2.1.41 cvector_replace

```
#define cvector_replace CONCAT(CVECTOR_T, _vect__replace)
```

Definition at line 172 of file cvector_interface.h.

4.2.1.42 cvector_replace_func

```
#define cvector_replace_func CONCAT(CVECTOR_T, _vect__replace_func)
```

Definition at line 173 of file cvector_interface.h.

4.2.1.43 cvector_reversed

```
#define cvector_reversed CONCAT(CVECTOR_T, _vect__reversed)
```

Definition at line 167 of file cvector_interface.h.

4.2.1.44 cvector_safeget

```
#define cvector_safeget CONCAT(CVECTOR_T, _vect__safeget)
```

Definition at line 162 of file cvector_interface.h.

4.2.1.45 cvector_safeset

```
#define cvector_safeset CONCAT(CVECTOR_T, _vect__safeset)
```

Definition at line 164 of file cvector_interface.h.

4.2.1.46 cvector_set

```
#define cvector_set CONCAT(CVECTOR_T, _vect__set)
```

Definition at line 163 of file cvector_interface.h.

4.2.1.47 CVECTOR_SHRINK_FACTOR

```
#define CVECTOR_SHRINK_FACTOR 0.5
```

Space factor used when a shrink operation is triggered. It means that the new space of the cvector will be SHRI↩ NK_FACTOR ∗ the current space.

Definition at line 81 of file cvector_interface.h.

4.2.1.48 CVECTOR_SHRINK_THRESHOLD

```
#define CVECTOR_SHRINK_THRESHOLD 0.5
```

Threshold from which the cvector will be shrank in a readjust operation. It means that if the current size of the cvector is under SHRINK_THRESHOLD ∗ its space, it will be shrank. Set to under 0 to prevent shrink during readjust operations.

Definition at line 73 of file cvector_interface.h.

4.2.1.49 cvector_slice

```
#define cvector_slice CONCAT(CVECTOR_T, _vect__slice)
```

Definition at line 179 of file cvector_interface.h.

4.2.1.50 cvector_slicetoarray

```
#define cvector_slicetoarray CONCAT(CVECTOR_T, _vect__slicetoarray)
```

Definition at line 180 of file cvector_interface.h.

4.2.1.51 cvector_sort

```
#define cvector_sort CONCAT(CVECTOR_T, _vect__sort)
```

Definition at line 174 of file cvector_interface.h.

4.2.1.52 CVECTOR_T

```
#define CVECTOR_T int
```

Type of the elements to hold in this instance of the cvector library. BE CAREFUL! The specified type must be a correct indentifier, since it will prefix any function of this cvector instance. For example #define CVECTOR_T int ∗ should be replaced with typedef int ∗ pint; #define CVECTOR_T pint

Definition at line 124 of file cvector_interface.h.

4.2.1.53 cvector_toarray

```
#define cvector_toarray CONCAT(CVECTOR_T, _vect__toarray)
```

Definition at line 171 of file cvector_interface.h.

4.2.1.54 hash_t

```
#define hash_t CVECTOR_HASH_T
```

Definition at line 140 of file cvector_interface.h.

4.2.1.55 index_t

```
#define index_t long
```

Definition at line 28 of file cvector_interface.h.

4.2.1.56 NOT_FOUND_INDEX

```
#define NOT_FOUND_INDEX ((index_t) (-1))
```

Definition at line 29 of file cvector_interface.h.

4.2.1.57 ROUND_INDEX

```
#define ROUND_INDEX(
              x ) ((index_t) (lrint(x)))
```

Definition at line 30 of file cvector_interface.h.

4.2.1.58 value_t

```
#define value_t CVECTOR_T
```

Definition at line 139 of file cvector_interface.h.

4.2.2 Typedef Documentation

4.2.2.1 cvector

```
typedef struct cvector cvector
```

Definition at line 185 of file cvector_interface.h.

4.2.3 Function Documentation

4.2.3.1 __cvector_extend()

```
void __cvector_extend (
            cvector * p_cvector )
```

Extends the specified cvector without any check about its size.

Parameters

p_cvector	a pointer to the cvector

Definition at line 51 of file cvector_core.h.

4.2.3.2 __cvector_setspace()

```
void __cvector_setspace (
            cvector * p_cvector,
            index_t new_space )
```

Sets space of the specified cvector to new_space

Parameters

p_cvector	a pointer to the cvector
new_space	the new space for the specified cvector

Definition at line 30 of file cvector_core.h.

4.2.3.3 __cvector_shrink()

```
void __cvector_shrink (
            cvector * p_cvector )
```

Shrinks the specified cvector without any check about its size.

Parameters

p_cvector	a pointer to the cvector

Definition at line 42 of file cvector_core.h.

4.2.3.4 cvector_add()

```
void cvector_add (
            cvector * p_cvector,
            value_t value )
```

Adds the specified element at the end of the cvector.

Parameters

p_cvector	a pointer to the cvector
value	the value to push at the end of the cvector

Definition at line 202 of file cvector_core.h.

4.2.3.5 cvector_addi()

```
void cvector_addi (
            cvector * p_cvector,
            value_t value,
            index_t index )
```

Adds the specified element a the position index in the cvector, and shift following elements to the right.

Parameters

p_cvector	a pointer to the cvector
value	the value to push a the position index in the cvector
index	the index where the specified value will be inserted

Definition at line 219 of file cvector_core.h.

4.2.3.6 cvector_addspace()

```
void cvector_addspace (
            cvector * p_cvector )
```

Adds space (according to the DEFAULT_ADDSPACE_FACTOR) to the specified cvector.

Parameters

p_cvector	a pointer to the cvector to extend.

Definition at line 75 of file cvector_core.h.

4.2.3.7 cvector_appendto()

```
void cvector_appendto (
            cvector * p_cvector,
            cvector * p_add )
```

Appends element of the cvector pointed by p_add at the end of the cvector pointed by p_cvector.

Parameters

p_cvector	a pointer to the cvector where elements will be appended
p_add	a pointer to the cvector containing elements to copy

Definition at line 440 of file cvector_core.h.

4.2.3.8 cvector_clear()

```
void cvector_clear (
            cvector * p_cvector )
```

Removes all elements of the cvector without changing its space (that is to say without calling cvector_readjust).

Parameters

p_cvector	a pointer to the cvector

Definition at line 330 of file cvector_core.h.

4.2.3.9 cvector_concat()

```
cvector* cvector_concat (
            cvector * p_cvector_1,
            cvector * p_cvector_2 )
```

Returns a new cvector which is the concatenation of the two specified cvectors

Parameters

p_cvector_1	a pointer to the first cvector to concatenate
p_cvector_2	a pointer to the first cvector to concatenate

Returns

a pointer to the resulting cvector

Definition at line 461 of file cvector_core.h.

4.2.3.10 cvector_drop()

```
value_t cvector_drop (
            cvector * p_cvector )
```

Removes the first element of the cvector. If the cvector is empty, prints an error anrd returns DEFAULT_VALUE.

Parameters

p_cvector	a pointer to the cvector

Returns

the remove (first) element, or DEFAULT_VALUE if an error occurs

Definition at line 321 of file cvector_core.h.

4.2.3.11 cvector_equal()

```
bool cvector_equal (
            cvector * p_cvector_1,
            cvector * p_cvector_2 )
```

Returns true iif both specified cvectors are equal.

Parameters

p_cvector_1	a pointer to the first cvector to test
p_cvector_2	a pointer to the second cvector to test

Returns

true if both specified cvectors are equal, false otherwise

Definition at line 517 of file cvector_core.h.

4.2.3.12 cvector_equal_func()

```
bool cvector_equal_func (
            cvector * p_cvector_1,
            cvector * p_cvector_2,
            bool(*)(value_t, value_t) equal_value )
```

Returns true iif both specified cvectors are equal according to the specified test function for values.

Parameters

p_cvector_1	a pointer to the first cvector to test
p_cvector_2	a pointer to the second cvector to test
equal_value	the test function for values. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

true if both specified cvectors are equal according to the test function, false otherwise

Definition at line 539 of file cvector_core.h.

4.2.3.13 cvector_free()

```
void cvector_free (
            cvector * p_cvector )
```

Frees the specified cvector.

Parameters

p_cvector	a pointer to the cvector to free

Definition at line 169 of file cvector_core.h.

4.2.3.14 cvector_free_func()

```
void cvector_free_func (
            cvector * p_vector,
            void(*)(value_t) free_value )
```

Applies the specified free function of each value of the cvector, and then frees it too.

Parameters

p_vector	a pointer to the cvector to free
free_value	the function to free each value of the cvector

Definition at line 180 of file cvector_core.h.

4.2.3.15 cvector_get()

```
value_t cvector_get (
            cvector * p_cvector,
            index_t index )
```

Returns the value at the specified index in the cvector. Prints an error message and returns DEFAULT_VALUE if the specified index is invalid.

Parameters

p_cvector	a pointer to the cvector
index	the index of the value to get

Returns

the desired value if the index is correct, DEFAULT_VALUE otherwise

Definition at line 341 of file cvector_core.h.

4.2.3.16 cvector_getsize()

```
index_t cvector_getsize (
            cvector * p_cvector )
```

Size getter. Returns the size of the cvector.

Parameters

p_cvector	a pointer to the cvector

Returns

the size of the cvector

Definition at line 193 of file cvector_core.h.

4.2.3.17 cvector_hash()

```
hash_t cvector_hash (
            cvector * p_cvector,
            hash_t(*)(value_t) hash_value )
```

Returns the hash of the specified cvector, using djb2 algorithm by Dan Bernstein, according to the specified hash function for values of the cvector.

Parameters

p_cvector	a pointer to the cvector to hash
hash_value	hash function for values of the cvector. Signature of the hash value function must be hash_t hash_value(value_t value)

Returns

the computed hash of the specified cvector

Definition at line 503 of file cvector_core.h.

4.2.3.18 cvector_in()

```
bool cvector_in (
            cvector * p_cvector,
            value_t value )
```

Returns true iif the specified value was found in the cvector.

Parameters

p_cvector	a pointer to the cvector
value	the value to found

Returns

true if the value was found, false otherwise

Definition at line 676 of file cvector_core.h.

4.2.3.19 cvector_in_func()

```
bool cvector_in_func (
            cvector * p_cvector,
            value_t value,
            bool(*)(value_t, value_t) equal_value )
```

Returns true iif the specified value was found in the cvector according to the specified test function.

Parameters

p_cvector	a pointer to the cvector
value	the value to found
equal_value	the test function to check equality between values. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

true if the value was found, false otherwise

Definition at line 690 of file cvector_core.h.

4.2.3.20 cvector_indexof()

```
index_t cvector_indexof (
            cvector * p_cvector,
            value_t value )
```

Returns the first index where the specified value is found in the cvector. If the value is not found, returns NOT_F↩ OUND_INDEX value.

Parameters

p_cvector	a pointer to the cvector
value	the value to found

Returns

the first index where the specified value was found, or NOT_FOUND_INDEX if it was not found

Definition at line 639 of file cvector_core.h.

4.2.3.21 cvector_indexof_func()

```
index_t cvector_indexof_func (
            cvector * p_cvector,
            value_t value,
            bool(*)(value_t, value_t) equal_value )
```

Returns the first index where the specified value is found, according to the specified test function. Il the value is not found, returns NOT_FOUND_INDEX value.

Parameters

p_cvector	a pointer to the cvector
value	the value to found
equal_value	the test function to check equality between values. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

the first index where the specified value was found, or NOT_FOUND_INDEX if it was not found

Definition at line 660 of file cvector_core.h.

4.2.3.22 cvector_insert()

```
void cvector_insert (
            cvector * p_cvector,
            value_t value )
```

Adds the specified value at the beginning of the cvector, and shift following elements to the right.

Parameters

p_cvector	a pointer to the cvector
value	the value to add at the beginning of the cvector

Definition at line 253 of file cvector_core.h.

4.2.3.23 cvector_new()

```
cvector* cvector_new ( )
```

Creates a new cvector which can hold at the beginning at least DEFAULT_INIT_SPACE elements.

Returns

a pointer to the new cvector

Definition at line 87 of file cvector_core.h.

4.2.3.24 cvector_new_copy()

```
cvector* cvector_new_copy (
            cvector * p_original )
```

Creates a new cvector which is a copy of the specified one.

Parameters

p_original	a pointer to the cvector to copy

Returns

a pointer to the new (clone) cvector

Definition at line 121 of file cvector_core.h.

4.2.3.25 cvector_new_copy_space()

```
cvector* cvector_new_copy_space (
            cvector * p_original,
            index_t space )
```

Creates a new cvector which is a copy of the specified one and which can hold at least space elements.

Parameters

p_original	a pointer to the cvector to copy
space	desired space for the new (clone) cvector. space must be greater or equal than the size of the original cvector

Returns

a pointer to the new (clone) cvector

Definition at line 142 of file cvector_core.h.

4.2.3.26 cvector_new_space()

```
cvector* cvector_new_space (
            index_t space )
```

Creates a new cvector which can hold at the beginning at least space elements.

Parameters

space	desired space for the new cvector

Returns

a pointer to the new cvector

Definition at line 102 of file cvector_core.h.

4.2.3.27 cvector_readjust()

```
void cvector_readjust (
            cvector * p_cvector )
```

Readjusts space of the specified cvector if needed, according to SHRINK_THRESHOLD and EXTEND_THRES↩ HOLD.

Parameters

p_cvector	a pointer to the cvector

Definition at line 61 of file cvector_core.h.

4.2.3.28 cvector_remove()

```
value_t cvector_remove (
            cvector * p_cvector )
```

Removes the last element of the cvector and returns it. If the cvector is empty, prints an error and returns DEFA↩ ULT_VALUE.

Parameters

p_cvector	a pointer to the cvector

Returns

The last value of the cvector if it is not empty, DEFAULT_VALUE otherwise

Definition at line 264 of file cvector_core.h.

4.2.3.29 cvector_removei()

```
value_t cvector_removei (
            cvector * p_cvector,
            index_t index )
```

Removes the element located at the specified index, and returns it. If the cvector is empty or if the index is incorrect, prints an error and returns DEFAULT_VALUE.

Parameters

p_cvector	a pointer to the cvector
index	the index where the element will be removed

Returns

the removed element or DEFAULT_VALUE if an error occurs

Definition at line 284 of file cvector_core.h.

4.2.3.30 cvector_replace()

```
bool cvector_replace (
            cvector * p_cvector,
            value_t original,
            value_t replacement )
```

Replace specified elements in the cvector and returns true if at least one change was made.

Parameters

p_cvector	a pointer to the cvector
original	original value to replace
replacement	replacement value for original

Returns

true if at least one replacement was made, false otherwise

Definition at line 577 of file cvector_core.h.

4.2.3.31 cvector_replace_func()

```
bool cvector_replace_func (
            cvector * p_cvector,
            value_t original,
            value_t replacement,
            bool(*)(value_t, value_t) equal_value )
```

Replace specified elements in the cvector and returns true if at least one change was made. Test between elements of the cvector and original are made with the specified function.

Parameters

p_cvector	a pointer to the cvector
original	original value to replace
replacement	replacement value for original
equal_value	test function used to compare cvector elements and original. Its signature must be bool equal_value(value_t value_1, value_t value_2)

Returns

true if at least one replacement was made, false otherwise

Definition at line 600 of file cvector_core.h.

4.2.3.32 cvector_reversed()

```
cvector* cvector_reversed (
            cvector * p_cvector )
```

Returns a cvector which contains the same elements as the specified one, but in a reversed order.

Parameters

p_cvector	a pointer to the original cvector

Returns

the resulting cvector, containing elements of the specified cvector in a reverse order

Definition at line 481 of file cvector_core.h.

4.2.3.33 cvector_safeget()

```
value_t cvector_safeget (
            cvector * p_cvector,
            index_t index )
```

Returns the value at the specified index in the cvector. Only prints a warning and returns DEFAULT_VALUE if the specified index is invalid.

Parameters

p_cvector	a pointer to the cvector
index	the index of the value to get

Returns

the desired value if the index is correct, DEFAULT_VALUE otherwise

Definition at line 364 of file cvector_core.h.

4.2.3.34 cvector_safeset()

```
void cvector_safeset (
            cvector * p_cvector,
            value_t value,
            index_t index )
```

Sets the value of the element located at the specified position. Only raises warning if the index is invalid, or extends the cvector to be able to set the value at the specified index.

Parameters

p_cvector	a pointer to the cvector
value	the value which will be inserted at the index position
index	the index where the value will be set

Definition at line 410 of file cvector_core.h.

4.2.3.35 cvector_set()

```
void cvector_set (
            cvector * p_cvector,
```

```
            value_t value,
            index_t index )
```

Sets the value of the element located at the specified index. Raises error if the specified index is invalid.

Parameters

p_cvector	a pointer to the cvector
value	the value which will be placed at the index position
index	the index where the value will be set

Definition at line 387 of file cvector_core.h.

4.2.3.36 cvector_slice()

```
cvector* cvector_slice (
            cvector * p_cvector,
            index_t from,
            index_t to,
            index_t step )
```

Returns the slice [|from:to[| of the specified cvector. Prints an error and return NULL if indexes are incorrect.

Parameters

p_cvector	a pointer to the cvector
from	index of the begin of the slice, included
to	index of the end of the slice, excluded
step	step of the slice

Returns

the corresponding (cvector) slice

Definition at line 707 of file cvector_core.h.

4.2.3.37 cvector_slicetoarray()

```
value_t* cvector_slicetoarray (
            cvector * p_cvector,
            index_t from,
            index_t to,
            index_t step )
```

Returns the slice [|from:to[| of the specified cvector as a c-style array. Prints an error and return NULL if indexes are incorrect.

Parameters

p_cvector	a pointer to the cvector
from	index of the begin of the slice, included
to	index of the end of the slice, excluded
step	step of the slice

Returns

the corresponding (c-style array) slice

Definition at line 755 of file cvector_core.h.

4.2.3.38 cvector_sort()

```
void cvector_sort (
            cvector * p_cvector,
            int(*)(const void *, const void *) comp_value )
```

Sorts the elements in the cvector according to the specified comparison function.

Parameters

p_cvector	a pointer to the cvector
comp_value	a comparison function which must have the signature int comp_value(const void ∗p_a, const void ∗p_b) and which must • return -1 if element a should be placed before element b • return 0 if element a and b could be placed at the same position • return 1 if element a should be placed after element b

Definition at line 624 of file cvector_core.h.

4.2.3.39 cvector_toarray()

```
value_t* cvector_toarray (
            cvector * p_cvector )
```

Returns a pointer to a c-style array holding the same elements as the specified cvector.

Parameters

p_cvector	a pointer to the cvector

Returns

a c-style malloc-ed array holding the same elements as the specified cvector, which must be freed after use

Definition at line 560 of file cvector_core.h.

Index