C-Array

0.1.3

Generated by Doxygen 1.8.11

Contents

Chapter 1

carray

Python-like list for C

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

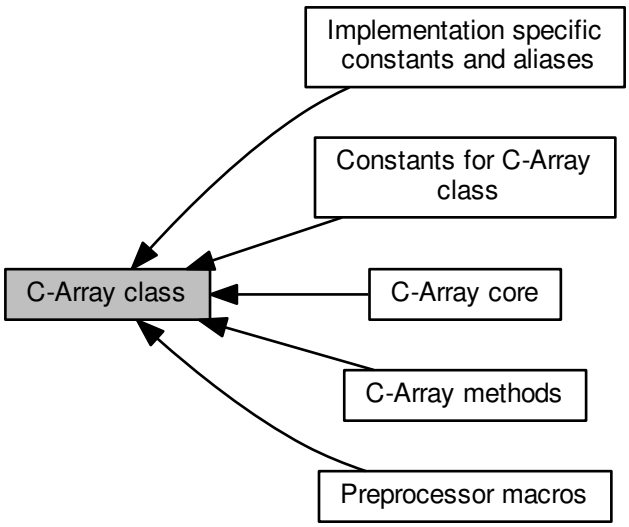
Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 C-Array class

Collaboration diagram for C-Array class:



Modules

- **Constants for C-Array class**
- **Implementation specific constants and aliases**
- **Preprocessor macros**
- **C-Array core**
- **C-Array methods**

Functions

- void **fatal_error** (char message[ ])
- **carray** ∗ **carray_new** ()
- **carray** ∗ **carray_new_ISC** (size_t init_space)
- **carray** ∗ **carray_new_CC** (**carray** ∗copy_carray)
- **carray** ∗ **carray_new_CISC** (**carray** ∗copy_carray, size_t init_space)
- void **carray_free** (**carray** ∗c, void(∗voidfunc)(type))
- size_t **carray_getsize** (**carray** ∗c)
- size_t **carray_getspace** (**carray** ∗c)
- void **carray_setspace** (**carray** ∗c, size_t new_space, void ∗∗ok)
- void **carray_addspace** (**carray** ∗c, void ∗∗ok)
- void **carray_safeset** (**carray** ∗c, int index, type value, void ∗∗ok)
- void **carray_append** (**carray** ∗c, **carray** ∗o, void ∗∗ok)
- size_t **carray_getreadposition** (**carray** ∗c)
- void **carray_setreadposition** (**carray** ∗c, int new_read_position, void ∗∗ok)
- size_t **carray_readingsremaining** (**carray** ∗c)
- type ∗ **carray_getarray** (**carray** ∗c)
- type **carray_read** (**carray** ∗c, void ∗∗ok)
- void **carray_push** (**carray** ∗c, type value)
- void **carray_ins** (**carray** ∗c, type value)
- type **carray_pop** (**carray** ∗c)
- void **carray_adjust** (**carray** ∗c, void ∗∗ok)
- void **carray_reverse** (**carray** ∗c)
- **carray** ∗ **carray_reversed_TF** (**carray** ∗c)
- **carray** ∗ **carray_concat_TF** (**carray** ∗a, **carray** ∗b)
- hashtype **carray_hashcode** (**carray** ∗c, hashtype(∗hashfunc)(type))
- bool **carray_equal** (**carray** ∗a, **carray** ∗b, bool(∗eqfunc)(type, type))
- char ∗ **carray_tostring_TF** (**carray** ∗c, char ∗(∗strfunc)(type), char ∗opener, char ∗closer, char ∗prefix, char ∗suffix)
- bool **carray_isempty** (**carray** ∗c)
- bool **carray_contains** (**carray** ∗c, type test_element, bool(∗eqfunc)(type, type))
- type ∗ **carray_toarray_TF** (**carray** ∗c)
- bool **carray_remove_elt** (**carray** ∗c, type test_element, bool(∗eqfunc)(type, type))
- void **carray_clear** (**carray** ∗c)
- type **carray_get** (**carray** ∗c, int index, void ∗∗ok)
- void **carray_set** (**carray** ∗c, int index, type new_value, void ∗∗ok)
- void **carray_add** (**carray** ∗c, int index, type new_value, void ∗∗ok)
- type **carray_remove** (**carray** ∗c, int index, void ∗∗ok)
- int **carray_indexof** (**carray** ∗c, type test_value, bool(∗eqfunc)(type, type))
- int **carray_lastindexof** (**carray** ∗c, type test_value, bool(∗eqfunc)(type, type))
- **carray** ∗ **carray_subcarray_TF** (**carray** ∗c, int from_index, int to_index, void ∗∗ok)
- **carray** ∗ **carray_subcarraystep_TF** (**carray** ∗c, int from_index, int to_index, int step, void ∗∗ok)
- type ∗ **carray_subarray_TF** (**carray** ∗c, int from_index, int to_index, void ∗∗ok)
- type ∗ **carray_subarraystep_TF** (**carray** ∗c, int from_index, int to_index, int step, void ∗∗ok)
- void **carray_free_obj** (type val)

5.1.1 Detailed Description

5.1.2 Function Documentation

5.1.2.1 void carray_add (carray ∗ *c,* int *index,* type *new_value,* void ∗∗ *ok*)

Adds the specified element at the specified index in the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where to insert the specified element. The element will be inserted at this exact position, and subsequent elements will be shifted.
new_value	element to insert at the specified index
ok	validation flag

Definition at line 725 of file carray.c.

5.1.2.2 void carray_addspace (carray * c, void ** ok)

Increases space allowed for the internal representation of the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
ok	validation flag

Definition at line 189 of file carray.c.

5.1.2.3 void carray_adjust (carray * c, void ** ok)

Enventually shrinks the size of the internal representation of the carray if free space exceed the threshold. ok will hold the carray address if a shrinking was made, and NULL otherwise.

Parameters

c	the carray
ok	validation flag

Definition at line 386 of file carray.c.

5.1.2.4 void carray_append (carray * c, carray * o, void ** ok)

Append the second specified carray at the end of the first one. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
o	the carray to put at the end of the first one
ok	validation flag

Definition at line 251 of file carray.c.

5.1.2.5 void carray_clear (carray * c)

Clears the content of the carray. Be careful, this function DOESN'T free the space allowed for the internal representation of the carray. Use carray_adjust for that.

Parameters

c	the carray

Definition at line 652 of file carray.c.

5.1.2.6 carray* carray_concat_TF (carray * a, carray * b)

Concatenates both specified carrays into a new one which must be freed after use.

Parameters

a	the first carray to concatenate
b	the second carray to concatenate

Returns

a concatenated version of carrays a and b

Definition at line 438 of file carray.c.

5.1.2.7 bool carray_contains (carray * c, type test_element, bool(*)(type, type) eqfunc)

Tests if the carray contains the test_element according to the specified equality function

Parameters

c	the carray
test_element	the element to be compared to the carray elements
eqfunc	the function to test equality between elements

Returns

true if the specified element is inside the carray, and false otherwise

Definition at line 598 of file carray.c.

5.1.2.8 bool carray_equal (carray * a, carray * b, bool(*)(type, type) eqfunc)

Returns true if both carrays are equal and false otherwise, according to the specified elements equality function.

Parameters

a	the first carray to test
b	the second carray to test
eqfunc	equality function to apply on two elements

Returns

true if both carrays are equal, false otherwise

Definition at line 473 of file carray.c.

5.1.2.9 void carray_free (carray * c, void(*)(type) voidfunc)

Destructor for carray. Frees the carray internal array representation and the carray itself. If voidfunc is not NULL, applies this function to each element of the carray before freeing the whole struct.

Parameters

c	the carray
voidfunc	a function to be applied on each element of the carray to free it; can be NULL

Definition at line 120 of file carray.c.

5.1.2.10 void carray_free_obj (type val)

Frees the specified primitive-wrapped element.

Parameters

val	primitive-wrapped element to free

Definition at line 1077 of file carray.c.

5.1.2.11 type carray_get (carray * c, int index, void ** ok)

Gets the element at the specified index in the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where element will be retrieved
ok	validation flag

Returns

the element at the specified index if index is correct; default type value otherwise

Definition at line 667 of file carray.c.

5.1.2.12 type* carray_getarray (carray * c)

Internal array getter.

Parameters

c	the carray

Returns

the internal representation of the carray

Definition at line 320 of file carray.c.

5.1.2.13 size_t carray_getreadposition (carray * c)

Read position getter.

Parameters

c	the carray

Returns

the current read position

Definition at line 276 of file carray.c.

5.1.2.14 size_t carray_getsize (carray * c)

Size getter.

Parameters

c	the carray

Returns

the size of the carray

Definition at line 138 of file carray.c.

5.1.2.15 size_t carray_getspace (carray * c)

Space getter.

Parameters

c	the carray

Returns

the space (in nb of elements) of the carray

Definition at line 148 of file carray.c.

5.1.2.16 hashtype carray_hashcode (carray * c, hashtype(*)(type) hashfunc)

Returns the hashcode of the carray according to the specified elements hash code function.

Parameters

c	the carray
hashfunc	hashcode function to apply on each element of the carray

Returns

the hashcode of the carray

Definition at line 455 of file carray.c.

5.1.2.17 int carray_indexof (carray * c, type test_value, bool(*)(type, type) eqfunc)

Returns the index of the specified element in the carray, and -1 if this element wasn't found. The carray is read left to right and this function returns the index of the first occurence met.

Parameters

c	the carray
test_value	the element to find in the carray
eqfunc	the function to test equality between elements

Returns

the index of the specified element if this one was found, -1 otherwise

Definition at line 813 of file carray.c.

5.1.2.18 void carray_ins (carray * c, type value)

Inserts the specified value at the beginning of the carray.

Parameters

c	the carray
value	the element to insert at the end of the carray

Definition at line 362 of file carray.c.

5.1.2.19 bool carray_isempty (carray * c)

Returns true if the carray is empty, false otherwise

Parameters

c	the carray

Returns

true if the carray is empty, false otherwise

Definition at line 584 of file carray.c.

5.1.2.20 int carray_lastindexof (carray * c, type test_value, bool(*)(type, type) eqfunc)

Returns the index of the specified element in the carray, and -1 if this element wasn't found. The carray is read right to left and this function returns the index of the first occurence met.

Parameters

c	the carray
test_value	the element to find in the carray
eqfunc	the function to test equality between elements

Returns

the index of the specified element if this one was found, -1 otherwise

Definition at line 838 of file carray.c.

5.1.2.21 carray* carray_new ( )

Constructor for carray. Returns a pointer to the created carray which must be freed after use.

Returns

a pointer to the created carray

Definition at line 37 of file carray.c.

5.1.2.22 carray* carray_new_CC (carray * *copy_carray*)

Copy constructor for carray. The created carray is the exact copy of the specified one with a bit more space than the length of the specified one. Returns a pointer to the created carray which must be freed after use.

Parameters

copy_carray	the carray to be copied

Returns

a pointer to the created carray

Definition at line 76 of file carray.c.

5.1.2.23 carray* carray_new_CISC (carray * *copy_carray,* size_t *init_space*)

Copy constructor for carray with init_space specified. This carray have at least init_space slots at instanciation, and hold the exact content of the specified carray. Returns a pointer to the created carray which must be freed after use; if init_space is shorter than the length of the carray to be copied, returns NULL.

Parameters

copy_carray	the carray to be copied
init_space	Initial space (number of elements) of the carray

Returns

a pointer to the created carray

Definition at line 97 of file carray.c.

5.1.2.24 carray* carray_new_ISC (size_t *init_space*)

Constructor for carray with specified init_space. This carray have at least init_space slots at instanciation. Returns a pointer to the created carray which must be freed after use; if init_space is not valid, returns NULL.

Parameters

init_space	Initial space (number of elements) of the carray

Returns

a pointer to the created carray

Definition at line 55 of file carray.c.

5.1.2.25 type carray_pop (carray * c)

Removes the last element of the carray and returns it.

Parameters

c	the carray

Returns

the last element of the carray

Definition at line 373 of file carray.c.

5.1.2.26 void carray_push (carray * c, type value)

Pushes the specified value at the end of the carray.

Parameters

c	the carray
value	the element to push at the end of the carray

Definition at line 351 of file carray.c.

5.1.2.27 type carray_read (carray * c, void ** ok)

Read method. Returns elt at the current read position and increases read position by 1. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
ok	validation flag

Returns

the element at the current read position; default type value if there's no element to be read

Definition at line 335 of file carray.c.

5.1.2.28 size_t carray_readingsremaining (carray * c)

Gets the number of read operation remaining before reaching the end of the carray.

Parameters

c	the carray

Returns

the number of read possible before the end of the carray

Definition at line 310 of file carray.c.

5.1.2.29 type carray_remove (carray * c, int index, void ** ok)

Removes the element at the specified index and returns it. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where element will be removed. The element will be removed at this exact position, and subsequent elements will be shifted.
ok	validation flag

Returns

the element which have been removed if the index is correct, default type value otherwise.

Definition at line 778 of file carray.c.

5.1.2.30 bool carray_remove_elt (carray * c, type test_element, bool(*)(type, type) eqfunc)

Removes the specified element from the carray if and only if this one is present.

Parameters

c	the carray
test_element	element to be removed if present
eqfunc	the function to test equality between elements

Returns

true if the specified element was in the carray and thus was removed, false if nothing was made.

Definition at line 634 of file carray.c.

5.1.2.31 void carray_reverse (carray * c)

Reverses the specified carray in-place.

Parameters

c	the carray

Definition at line 402 of file carray.c.

5.1.2.32 carray* carray_reversed_TF (carray * c)

Returns a reversed copy of the specified carray which must be freed after use.

Parameters

c	the carray

Returns

a reversed copy of the specified carray

Definition at line 419 of file carray.c.

5.1.2.33 void carray_safeset (carray * c, int index, type value, void ** ok)

Used to set an element somewhere in the carray even if this cell is not already used by the carray, for example, to set carray[8] with a length-4 carray. Unused cells until the specified one are set with the default type value. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where to put the specified value; can be larger than the carray size
value	value to be put at the specified index
ok	validation flag

Definition at line 207 of file carray.c.

5.1.2.34 void carray_set (carray * c, int index, type new_value, void ** ok)

Sets the element at the specified index in the carray to the specified value. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray

Parameters

index	index where element will be set
new_value	new value for this element
ok	validation flag

Definition at line 697 of file carray.c.

5.1.2.35 void carray_setreadposition (carray * c, int new_read_position, void ** ok)

Read position setter. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
new_read_position	new read position
ok	validation flag

Definition at line 289 of file carray.c.

5.1.2.36 void carray_setspace (carray * c, size_t new_space, void ** ok)

Space setter. Can be used to modify the space allowed for the internal representation of the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
new_space	new space (in nb of elements) for the carray
validation	flag

Definition at line 162 of file carray.c.

5.1.2.37 type* carray_subarray_TF (carray * c, int from_index, int to_index, void ** ok)

Returns a smaller vanilla array which holds values from index "from_index" (included) to index "to_index" (excluded). If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
ok	validation flag

Returns

the sub-vanilla array if the indices are correct, NULL otherwise

Definition at line 977 of file carray.c.

5.1.2.38 type* carray_subarraystep_TF (carray * *c*, int *from_index*, int *to_index*, int *step*, void ** *ok*)

Returns a smaller vanilla array which holds values from index "from_index" (included) to index "to_index" (excluded) according to the specified step. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
step	item selecting step
ok	validation flag

Returns

the sub-vanilla array if the indices are correct, NULL otherwise

Definition at line 1019 of file carray.c.

5.1.2.39 carray* carray_subcarray_TF (carray * *c*, int *from_index*, int *to_index*, void ** *ok*)

Returns a smaller carray which holds values from index "from_index" (included) to index "to_index" (excluded). If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
ok	validation flag

Returns

the sub-carray if the indices are correct, NULL otherwise

Definition at line 864 of file carray.c.

5.1.2.40 carray* carray_subcarraystep_TF (carray * *c*, int *from_index*, int *to_index*, int *step*, void ** *ok*)

Returns a smaller carray which holds values from index "from_index" (included) to index "to_index" (excluded) according to the specified step. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
step	item selecting step
ok	validation flag

Returns

the sub-carray if the indices are correct, NULL otherwise

Definition at line 908 of file carray.c.

5.1.2.41 type* carray_toarray_TF (carray * *c*)

Returns a vanilla array version of the carray.

Parameters

c	the carray

Returns

a vanilla array version of the carray

Definition at line 618 of file carray.c.

5.1.2.42 char* carray_tostring_TF (carray * *c*, char *(*)(type) *strfunc*, char * *opener*, char * *closer*, char * *prefix*, char * *suffix*)

Returns a representation of the specified carray.

Parameters

c	the carray
strfunc	the function to apply on each element to convert it into string
opener	opening string
closer	closing string
prefix	prefix string added before each cell except for the first one
suffix	suffix string added after each cell except for the lats one

Returns

a string representing the carray

Definition at line 500 of file carray.c.

5.1.2.43 void fatal_error (char *message[ ]*)

Prints an error message

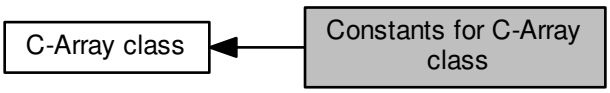
Parameters

message	message to print

Definition at line 27 of file carray.c.

5.2 Constants for C-Array class

Collaboration diagram for Constants for C-Array class:



Macros

- #define **DEFAULT_SPACE_INIT_FLAT** 8
- #define **DEFAULT_SPACE_INIT_PERCENT** 1.25
- #define **DEFAULT_SPACE_INCR** 2.0
- #define **DEFAULT_SHRINK_THRESHOLD** 2.5
- #define **DEFAULT_SHRINK_PERCENT** 1.25

Typedefs

- typedef struct **carray carray**

5.2.1 Detailed Description

5.2.2 Macro Definition Documentation

5.2.2.1 #define DEFAULT_SHRINK_PERCENT 1.25

Space factor used when the carray is shrinked.

Definition at line 50 of file carray.h.

5.2.2.2 #define DEFAULT_SHRINK_THRESHOLD 2.5

Space threshold from which the carray is shrinked.

Definition at line 45 of file carray.h.

5.2.2.3 #define DEFAULT_SPACE_INCR 2.0

Space factor used when a carray becomes too short.

Definition at line 40 of file carray.h.

5.2.2.4 #define DEFAULT_SPACE_INIT_FLAT 8

Initial space for the internal representation of a carray.

Definition at line 30 of file carray.h.

5.2.2.5 #define DEFAULT_SPACE_INIT_PERCENT 1.25

Initial space factor used when a carray is created from an existing one.

Definition at line 35 of file carray.h.

5.2.3 Typedef Documentation

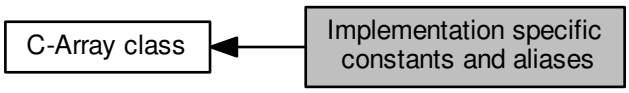
5.2.3.1 typedef struct carray carray

Alias for carray struct.

Definition at line 55 of file carray.h.

5.3 Implementation specific constants and aliases

Collaboration diagram for Implementation specific constants and aliases:



Macros

- #define **PRIME_CST** 31
- #define **DEFAULT_TYPE_VALUE** NULL

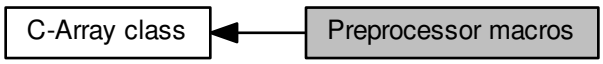
Typedefs

- typedef unsigned int **hashtype**
- typedef void * **type**

5.3.1 Detailed Description

5.4 Preprocessor macros

Collaboration diagram for Preprocessor macros:



Macros

- #define **of_Char** ∗(char∗)
- #define **of_sChar** ∗(signed char∗)
- #define **of_uChar** ∗(unsigned char∗)
- #define **of_Short** ∗(short∗)
- #define **of_sShort** ∗(signed short∗)
- #define **of_uShort** ∗(unsigned short∗)
- #define **of_Int** ∗(int∗)
- #define **of_sInt** ∗(signed int∗)
- #define **of_uInt** ∗(unsigned int∗)
- #define **of_Long** ∗(long∗)
- #define **of_lLong** ∗(long long∗)
- #define **of_sLong** ∗(signed long∗)
- #define **of_slLong** ∗(signed long long∗)
- #define **of_uLong** ∗(unsigned long∗)
- #define **of_ulLong** ∗(unsigned long long∗)
- #define **of_Float** ∗(float∗)
- #define **of_Double** ∗(double∗)
- #define **of_lDouble** ∗(long double∗)
- #define **of_Bool** ∗(bool∗)
- #define **of_String** (char∗)
- #define **Char**(expr)
- #define **sChar**(expr)
- #define **uChar**(expr)
- #define **Short**(expr)
- #define **sShort**(expr)
- #define **uShort**(expr)
- #define **Int**(expr)
- #define **sInt**(expr)
- #define **uInt**(expr)
- #define **Long**(expr)
- #define **lLong**(expr)
- #define **sLong**(expr)
- #define **slLong**(expr)
- #define **uLong**(expr)
- #define **ulLong**(expr)
- #define **Float**(expr)
- #define **Double**(expr)
- #define **lDouble**(expr)
- #define **Bool**(expr)
- #define **String**(expr)
- #define **free_Obj** &**carray_free_obj**

Functions

- void **carray_free_obj** (type)

Variables

- char * **Char_holder**
- signed char * **sChar_holder**
- unsigned char * **uChar_holder**
- short * **Short_holder**
- signed short * **sShort_holder**
- unsigned short * **uShort_holder**
- int * **Int_holder**
- signed int * **sInt_holder**
- unsigned int * **uInt_holder**
- long * **Long_holder**
- long long * **lLong_holder**
- signed long * **sLong_holder**
- signed long long * **slLong_holder**
- unsigned long * **uLong_holder**
- unsigned long long * **ulLong_holder**
- float * **Float_holder**
- double * **Double_holder**
- long double * **lDouble_holder**
- bool * **Bool_holder**
- char * **String_holder**

5.4.1 Detailed Description

5.4.2 Macro Definition Documentation

5.4.2.1 #define Bool(*expr*)

Value:

```
(Bool_holder = malloc(sizeof(bool)), \
     *Bool_holder = (expr), Bool_holder)
```

Definition at line 189 of file carray.h.

5.4.2.2 #define Char(*expr*)

Value:

```
(Char_holder = malloc(sizeof(char)), \
     *Char_holder = (expr), Char_holder)
```

Definition at line 99 of file carray.h.

5.4.2.3 #define Double(*expr*)

Value:

```
(Double_holder = malloc(sizeof(double)), \
     *Double_holder = (expr), Double_holder)
```

Definition at line 179 of file carray.h.

5.4.2.4 #define Float(*expr*)

Value:

```
(Float_holder = malloc(sizeof(float)), \
     *Float_holder = (expr), Float_holder)
```

Definition at line 174 of file carray.h.

5.4.2.5 #define Int(*expr*)

Value:

```
(Int_holder = malloc(sizeof(int)), \
     *Int_holder = (expr), Int_holder)
```

Definition at line 129 of file carray.h.

5.4.2.6 #define lDouble(*expr*)

Value:

```
(lDouble_holder = malloc(sizeof(long double)), \
     *lDouble_holder = (expr), lDouble_holder)
```

Definition at line 184 of file carray.h.

5.4.2.7 #define lLong(*expr*)

Value:

```
(lLong_holder = malloc(sizeof(long long)), \
     *lLong_holder = (expr), lLong_holder)
```

Definition at line 149 of file carray.h.

5.4.2.8 #define Long(*expr*)

Value:

```
(Long_holder = malloc(sizeof(long)), \
     *Long_holder = (expr), Long_holder)
```

Definition at line 144 of file carray.h.

5.4.2.9 #define sChar(*expr*)

Value:

```
(sChar_holder = malloc(sizeof(signed char)), \
     *sChar_holder = (expr), sChar_holder)
```

Definition at line 104 of file carray.h.

5.4.2.10 #define Short(*expr*)

Value:

```
(Short_holder = malloc(sizeof(short)), \
     *Short_holder = (expr), Short_holder)
```

Definition at line 114 of file carray.h.

5.4.2.11 #define sInt(*expr*)

Value:

```
(sInt_holder = malloc(sizeof(signed int)), \
     *sInt_holder = (expr), sInt_holder)
```

Definition at line 134 of file carray.h.

5.4.2.12 #define slLong(*expr*)

Value:

```
(slLong_holder = malloc(sizeof(signed long long)), \
     *slLong_holder = (expr), slLong_holder)
```

Definition at line 159 of file carray.h.

5.4.2.13 #define sLong(*expr*)

Value:

```
(sLong_holder = malloc(sizeof(signed long)), \
     *sLong_holder = (expr), sLong_holder)
```

Definition at line 154 of file carray.h.

5.4.2.14 #define sShort(*expr*)

Value:

```
(sShort_holder = malloc(sizeof(signed short)), \
     *sShort_holder = (expr), sShort_holder)
```

Definition at line 119 of file carray.h.

5.4.2.15 #define String(*expr*)

Value:

```
(String_holder = malloc(sizeof(expr) / sizeof(char)), \
     strcpy(String_holder, (expr)), \
     String_holder)
```

Definition at line 194 of file carray.h.

5.4.2.16 #define uChar(*expr*)

Value:

```
(uChar_holder = malloc(sizeof(unsigned char)), \
     *uChar_holder = (expr), uChar_holder)
```

Definition at line 109 of file carray.h.

5.4.2.17 #define uInt(*expr*)

Value:

```
(uInt_holder = malloc(sizeof(unsigned int)), \
     *uInt_holder = (expr), uInt_holder)
```

Definition at line 139 of file carray.h.

5.4.2.18 #define ulLong(*expr*)

Value:

```
(ulLong_holder = malloc(sizeof(unsigned long long)), \
     *ulLong_holder = (expr), ulLong_holder)
```

Definition at line 169 of file carray.h.

5.4.2.19 #define uLong(*expr*)

Value:

```
(uLong_holder = malloc(sizeof(unsigned long)), \
     *uLong_holder = (expr), uLong_holder)
```

Definition at line 164 of file carray.h.

5.4.2.20 #define uShort(*expr*)

Value:

```
(uShort_holder = malloc(sizeof(unsigned short)), \
     *uShort_holder = (expr), uShort_holder)
```

Definition at line 124 of file carray.h.

5.4.3 Function Documentation

5.4.3.1 void carray_free_obj (type *val*)

Frees the specified primitive-wrapped element.

Parameters

val	primitive-wrapped element to free

Definition at line 1077 of file carray.c.

5.5 C-Array core

Collaboration diagram for C-Array core:

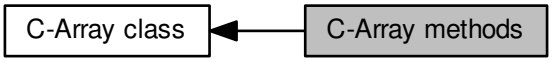
Data Structures

- struct **carray**

5.5.1 Detailed Description

5.6 C-Array methods

Collaboration diagram for C-Array methods:



Functions

- **carray** * **carray_new** ()
- **carray** * **carray_new_ISC** (size_t)
- **carray** * **carray_new_CC** (**carray** *)
- **carray** * **carray_new_CISC** (**carray** *, size_t)
- void **carray_free** (**carray** *, void(type))
- size_t **carray_getsize** (**carray** *)
- void **carray_clear** (**carray** *)
- bool **carray_isempty** (**carray** *)
- size_t **carray_getspace** (**carray** *)
- void **carray_setspace** (**carray** *, size_t, void **)
- void **carray_adjust** (**carray** *, void **)
- size_t **carray_getreadposition** (**carray** *)
- size_t **carray_readingsremaining** (**carray** *)
- void **carray_setreadposition** (**carray** *, int, void **)
- type * **carray_getarray** (**carray** *)
- type **carray_read** (**carray** *, void **ok)
- void **carray_reverse** (**carray** *)
- **carray** * **carray_reversed_TF** (**carray** *)
- void **carray_append** (**carray** *, **carray** *, void **)
- **carray** * **carray_concat_TF** (**carray** *, **carray** *)
- char * **carray_tostring_TF** (**carray** *, char *(type), char *, char *, char *, char *)
- hashtype **carray_hashcode** (**carray** *, hashtype(type))
- bool **carray_equal** (**carray** *, **carray** *, bool(type, type))
- int **carray_indexof** (**carray** *, type, bool(type, type))
- int **carray_lastindexof** (**carray** *, type, bool(type, type))
- bool **carray_contains** (**carray** *, type, bool(type, type))
- **carray** * **carray_subcarray_TF** (**carray** *, int, int, void **)
- **carray** * **carray_subcarraystep_TF** (**carray** *, int, int, int, void **)
- type * **carray_subarray_TF** (**carray** *, int, int, void **)
- type * **carray_subarraystep_TF** (**carray** *, int, int, int, void **)
- type * **carray_toarray_TF** (**carray** *)
- type **carray_get** (**carray** *, int, void **)
- void **carray_add** (**carray** *, int, type, void **)
- void **carray_push** (**carray** *, type)
- void **carray_ins** (**carray** *, type)
- void **carray_set** (**carray** *, int, type, void **)
- void **carray_safeset** (**carray** *, int, type, void **)
- type **carray_remove** (**carray** *, int, void **)
- type **carray_pop** (**carray** *)
- bool **carray_remove_elt** (**carray** *, type, bool(type, type))

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 void carray_add (carray * c, int index, type new_value, void ** ok)

Adds the specified element at the specified index in the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where to insert the specified element. The element will be inserted at this exact position, and subsequent elements will be shifted.
new_value	element to insert at the specified index
ok	validation flag

Definition at line 725 of file carray.c.

5.6.2.2 void carray_adjust (carray * c, void ** ok)

Enventually shrinks the size of the internal representation of the carray if free space exceed the threshold. ok will hold the carray address if a shrinking was made, and NULL otherwise.

Parameters

c	the carray
ok	validation flag

Definition at line 386 of file carray.c.

5.6.2.3 void carray_append (carray * c, carray * o, void ** ok)

Append the second specified carray at the end of the first one. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
o	the carray to put at the end of the first one
ok	validation flag

Definition at line 251 of file carray.c.

5.6.2.4 void carray_clear (carray * c)

Clears the content of the carray. Be careful, this function DOESN'T free the space allowed for the internal representation of the carray. Use carray_adjust for that.

Parameters

c	the carray

Definition at line 652 of file carray.c.

5.6.2.5 carray* carray_concat_TF (carray * a, carray * b)

Concatenates both specified carrays into a new one which must be freed after use.

Parameters

a	the first carray to concatenate
b	the second carray to concatenate

Returns

a concatenated version of carrays a and b

Definition at line 438 of file carray.c.

5.6.2.6 type carray_get (carray * c, int index, void ** ok)

Gets the element at the specified index in the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where element will be retrieved
ok	validation flag

Returns

the element at the specified index if index is correct; default type value otherwise

Definition at line 667 of file carray.c.

5.6.2.7 type* carray_getarray (carray * c)

Internal array getter.

Parameters

c	the carray

Returns

the internal representation of the carray

Definition at line 320 of file carray.c.

5.6.2.8 size_t carray_getreadposition (carray * c)

Read position getter.

Parameters

c	the carray

Returns

the current read position

Definition at line 276 of file carray.c.

5.6.2.9 size_t carray_getsize (carray * c)

Size getter.

Parameters

c	the carray

Returns

the size of the carray

Definition at line 138 of file carray.c.

5.6.2.10 size_t carray_getspace (carray * c)

Space getter.

Parameters

c	the carray

Returns

the space (in nb of elements) of the carray

Definition at line 148 of file carray.c.

5.6.2.11 void carray_ins (carray * c, type value)

Inserts the specified value at the beginning of the carray.

Parameters

c	the carray
value	the element to insert at the end of the carray

Definition at line 362 of file carray.c.

5.6.2.12 bool carray_isempty (carray * c)

Returns true if the carray is empty, false otherwise

Parameters

c	the carray

Returns

true if the carray is empty, false otherwise

Definition at line 584 of file carray.c.

5.6.2.13 carray* carray_new ( )

Constructor for carray. Returns a pointer to the created carray which must be freed after use.

Returns

a pointer to the created carray

Definition at line 37 of file carray.c.

5.6.2.14 carray* carray_new_CC (carray * copy_carray)

Copy constructor for carray. The created carray is the exact copy of the specified one with a bit more space than the length of the specified one. Returns a pointer to the created carray which must be freed after use.

Parameters

copy_carray	the carray to be copied

Returns

a pointer to the created carray

Definition at line 76 of file carray.c.

5.6.2.15 carray* carray_new_CISC (carray * *copy_carray,* size_t *init_space*)

Copy constructor for carray with init_space specified. This carray have at least init_space slots at instanciation, and hold the exact content of the specified carray. Returns a pointer to the created carray which must be freed after use; if init_space is shorter than the length of the carray to be copied, returns NULL.

Parameters

copy_carray	the carray to be copied
init_space	Initial space (number of elements) of the carray

Returns

a pointer to the created carray

Definition at line 97 of file carray.c.

5.6.2.16 carray* carray_new_ISC (size_t *init_space*)

Constructor for carray with specified init_space. This carray have at least init_space slots at instanciation. Returns a pointer to the created carray which must be freed after use; if init_space is not valid, returns NULL.

Parameters

init_space	Initial space (number of elements) of the carray

Returns

a pointer to the created carray

Definition at line 55 of file carray.c.

5.6.2.17 type carray_pop (carray * *c*)

Removes the last element of the carray and returns it.

Parameters

c	the carray

Returns

the last element of the carray

Definition at line 373 of file carray.c.

5.6.2.18 void carray_push (carray * c, type value)

Pushes the specified value at the end of the carray.

Parameters

c	the carray
value	the element to push at the end of the carray

Definition at line 351 of file carray.c.

5.6.2.19 type carray_read (carray * c, void ** ok)

Read method. Returns elt at the current read position and increases read position by 1. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
ok	validation flag

Returns

the element at the current read position; default type value if there's no element to be read

Definition at line 335 of file carray.c.

5.6.2.20 size_t carray_readingsremaining (carray * c)

Gets the number of read operation remaining before reaching the end of the carray.

Parameters

c	the carray

Returns

the number of read possible before the end of the carray

Definition at line 310 of file carray.c.

5.6.2.21 type carray_remove (carray * c, int index, void ** ok)

Removes the element at the specified index and returns it. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where element will be removed. The element will be removed at this exact position, and subsequent elements will be shifted.
ok	validation flag

Returns

the element which have been removed if the index is correct, default type value otherwise.

Definition at line 778 of file carray.c.

5.6.2.22 void carray_reverse (carray * c)

Reverses the specified carray in-place.

Parameters

c	the carray

Definition at line 402 of file carray.c.

5.6.2.23 carray* carray_reversed_TF (carray * c)

Returns a reversed copy of the specified carray which must be freed after use.

Parameters

c	the carray

Returns

a reversed copy of the specified carray

Definition at line 419 of file carray.c.

5.6.2.24 void carray_safeset (carray * c, int index, type value, void ** ok)

Used to set an element somewhere in the carray even if this cell is not already used by the carray, for example, to set carray[8] with a length-4 carray. Unused cells until the specified one are set with the default type value. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where to put the specified value; can be larger than the carray size
value	value to be put at the specified index
ok	validation flag

Definition at line 207 of file carray.c.

5.6.2.25 void carray_set (carray * c, int index, type new_value, void ** ok)

Sets the element at the specified index in the carray to the specified value. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
index	index where element will be set
new_value	new value for this element
ok	validation flag

Definition at line 697 of file carray.c.

5.6.2.26 void carray_setreadposition (carray * c, int new_read_position, void ** ok)

Read position setter. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
new_read_position	new read position
ok	validation flag

Definition at line 289 of file carray.c.

5.6.2.27 void carray_setspace (carray * c, size_t new_space, void ** ok)

Space setter. Can be used to modify the space allowed for the internal representation of the carray. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
new_space	new space (in nb of elements) for the carray
validation	flag

Definition at line 162 of file carray.c.

5.6.2.28 type* carray_subarray_TF (carray * c, int from_index, int to_index, void ** ok)

Returns a smaller vanilla array which holds values from index "from_index" (included) to index "to_index" (excluded). If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
ok	validation flag

Returns

the sub-vanilla array if the indices are correct, NULL otherwise

Definition at line 977 of file carray.c.

5.6.2.29 type* carray_subarraystep_TF (carray * c, int from_index, int to_index, int step, void ** ok)

Returns a smaller vanilla array which holds values from index "from_index" (included) to index "to_index" (excluded) according to the specified step. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
step	item selecting step
ok	validation flag

Returns

the sub-vanilla array if the indices are correct, NULL otherwise

Definition at line 1019 of file carray.c.

5.6.2.30 carray* carray_subcarray_TF (carray * c, int from_index, int to_index, void ** ok)

Returns a smaller carray which holds values from index "from_index" (included) to index "to_index" (excluded). If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
ok	validation flag

Returns

the sub-carray if the indices are correct, NULL otherwise

Definition at line 864 of file carray.c.

5.6.2.31 carray* carray_subcarraystep_TF (carray * c, int from_index, int to_index, int step, void ** ok)

Returns a smaller carray which holds values from index "from_index" (included) to index "to_index" (excluded) according to the specified step. If params are correct and if the operation works, ok will hold the carray address; otherwise, it will hold NULL.

Parameters

c	the carray
from_index	beginning index
to_index	ending index
step	item selecting step
ok	validation flag

Returns

the sub-carray if the indices are correct, NULL otherwise

Definition at line 908 of file carray.c.

5.6.2.32 type* carray_toarray_TF (carray * c)

Returns a vanilla array version of the carray.

Parameters

c	the carray

Returns

a vanilla array version of the carray

Definition at line 618 of file carray.c.

Chapter 6

Data Structure Documentation

6.1 carray Struct Reference

`#include <carray.h>`

Data Fields

- type * **_array**
- size_t **_size**
- size_t **_space**
- size_t **_read_position**

6.1.1 Detailed Description

Carray class implementation. Provides Python-like list for C.

Definition at line 216 of file carray.h.

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

- **carray.h**

Chapter 7

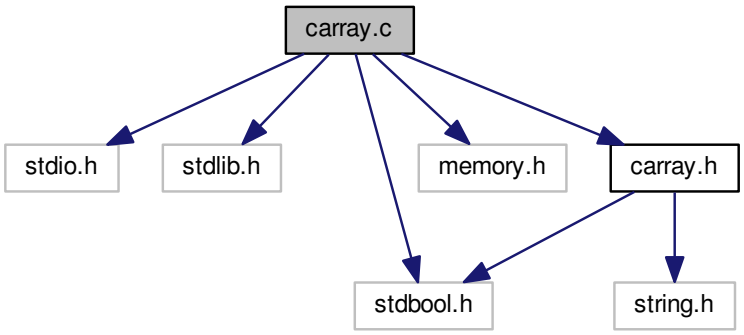
File Documentation

7.1 carray.c File Reference

Implementation of the carray class. Contains all the function implementations and documentation.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <memory.h>
#include "carray.h"
```

Include dependency graph for carray.c:



Functions

- void **fatal_error** (char message[ ])
- **carray** ∗ **carray_new** ()
- **carray** ∗ **carray_new_ISC** (size_t init_space)
- **carray** ∗ **carray_new_CC** (**carray** ∗copy_carray)
- **carray** ∗ **carray_new_CISC** (**carray** ∗copy_carray, size_t init_space)
- void **carray_free** (**carray** ∗c, void(∗voidfunc)(type))

- size_t **carray_getsize** (**carray** ∗c)
- size_t **carray_getspace** (**carray** ∗c)
- void **carray_setspace** (**carray** ∗c, size_t new_space, void ∗∗ok)
- void **carray_addspace** (**carray** ∗c, void ∗∗ok)
- void **carray_safeset** (**carray** ∗c, int index, type value, void ∗∗ok)
- void **carray_append** (**carray** ∗c, **carray** ∗o, void ∗∗ok)
- size_t **carray_getreadposition** (**carray** ∗c)
- void **carray_setreadposition** (**carray** ∗c, int new_read_position, void ∗∗ok)
- size_t **carray_readingsremaining** (**carray** ∗c)
- type ∗ **carray_getarray** (**carray** ∗c)
- type **carray_read** (**carray** ∗c, void ∗∗ok)
- void **carray_push** (**carray** ∗c, type value)
- void **carray_ins** (**carray** ∗c, type value)
- type **carray_pop** (**carray** ∗c)
- void **carray_adjust** (**carray** ∗c, void ∗∗ok)
- void **carray_reverse** (**carray** ∗c)
- **carray** ∗ **carray_reversed_TF** (**carray** ∗c)
- **carray** ∗ **carray_concat_TF** (**carray** ∗a, **carray** ∗b)
- hashtype **carray_hashcode** (**carray** ∗c, hashtype(∗hashfunc)(type))
- bool **carray_equal** (**carray** ∗a, **carray** ∗b, bool(∗eqfunc)(type, type))
- char ∗ **carray_tostring_TF** (**carray** ∗c, char ∗(∗strfunc)(type), char ∗opener, char ∗closer, char ∗prefix, char ∗suffix)
- bool **carray_isempty** (**carray** ∗c)
- bool **carray_contains** (**carray** ∗c, type test_element, bool(∗eqfunc)(type, type))
- type ∗ **carray_toarray_TF** (**carray** ∗c)
- bool **carray_remove_elt** (**carray** ∗c, type test_element, bool(∗eqfunc)(type, type))
- void **carray_clear** (**carray** ∗c)
- type **carray_get** (**carray** ∗c, int index, void ∗∗ok)
- void **carray_set** (**carray** ∗c, int index, type new_value, void ∗∗ok)
- void **carray_add** (**carray** ∗c, int index, type new_value, void ∗∗ok)
- type **carray_remove** (**carray** ∗c, int index, void ∗∗ok)
- int **carray_indexof** (**carray** ∗c, type test_value, bool(∗eqfunc)(type, type))
- int **carray_lastindexof** (**carray** ∗c, type test_value, bool(∗eqfunc)(type, type))
- **carray** ∗ **carray_subcarray_TF** (**carray** ∗c, int from_index, int to_index, void ∗∗ok)
- **carray** ∗ **carray_subcarraystep_TF** (**carray** ∗c, int from_index, int to_index, int step, void ∗∗ok)
- type ∗ **carray_subarray_TF** (**carray** ∗c, int from_index, int to_index, void ∗∗ok)
- type ∗ **carray_subarraystep_TF** (**carray** ∗c, int from_index, int to_index, int step, void ∗∗ok)
- void **carray_free_obj** (type val)

7.1.1 Detailed Description

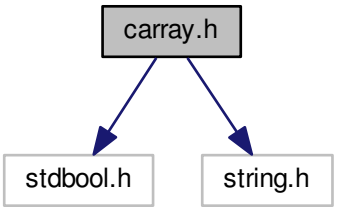
Implementation of the carray class. Contains all the function implementations and documentation.

7.2 carray.h File Reference

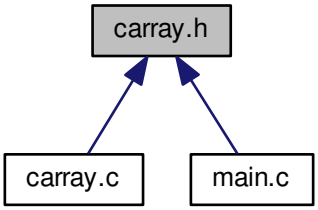
Header of the carray class. Contains all function declarations and preprocessor directives.

```
#include <stdbool.h>
#include <string.h>
```

Include dependency graph for carray.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct **carray**

Macros

- #define **DEFAULT_SPACE_INIT_FLAT** 8
- #define **DEFAULT_SPACE_INIT_PERCENT** 1.25
- #define **DEFAULT_SPACE_INCR** 2.0
- #define **DEFAULT_SHRINK_THRESHOLD** 2.5
- #define **DEFAULT_SHRINK_PERCENT** 1.25
- #define **PRIME_CST** 31
- #define **DEFAULT_TYPE_VALUE** NULL
- #define **of_Char** ∗(char∗)
- #define **of_sChar** ∗(signed char∗)
- #define **of_uChar** ∗(unsigned char∗)
- #define **of_Short** ∗(short∗)
- #define **of_sShort** ∗(signed short∗)

- #define **of_uShort** ∗(unsigned short∗)
- #define **of_Int** ∗(int∗)
- #define **of_sInt** ∗(signed int∗)
- #define **of_uInt** ∗(unsigned int∗)
- #define **of_Long** ∗(long∗)
- #define **of_lLong** ∗(long long∗)
- #define **of_sLong** ∗(signed long∗)
- #define **of_slLong** ∗(signed long long∗)
- #define **of_uLong** ∗(unsigned long∗)
- #define **of_ulLong** ∗(unsigned long long∗)
- #define **of_Float** ∗(float∗)
- #define **of_Double** ∗(double∗)
- #define **of_lDouble** ∗(long double∗)
- #define **of_Bool** ∗(bool∗)
- #define **of_String** (char∗)
- #define **Char**(expr)
- #define **sChar**(expr)
- #define **uChar**(expr)
- #define **Short**(expr)
- #define **sShort**(expr)
- #define **uShort**(expr)
- #define **Int**(expr)
- #define **sInt**(expr)
- #define **uInt**(expr)
- #define **Long**(expr)
- #define **lLong**(expr)
- #define **sLong**(expr)
- #define **slLong**(expr)
- #define **uLong**(expr)
- #define **ulLong**(expr)
- #define **Float**(expr)
- #define **Double**(expr)
- #define **lDouble**(expr)
- #define **Bool**(expr)
- #define **String**(expr)
- #define **free_Obj** &**carray_free_obj**

Typedefs

- typedef struct **carray carray**
- typedef unsigned int **hashtype**
- typedef void ∗ **type**

Functions

- void **carray_free_obj** (type)
- **carray** ∗ **carray_new** ()
- **carray** ∗ **carray_new_ISC** (size_t)
- **carray** ∗ **carray_new_CC** (**carray** ∗)
- **carray** ∗ **carray_new_CISC** (**carray** ∗, size_t)
- void **carray_free** (**carray** ∗, void(type))
- size_t **carray_getsize** (**carray** ∗)
- void **carray_clear** (**carray** ∗)

- bool **carray_isempty** (**carray** ∗)
- size_t **carray_getspace** (**carray** ∗)
- void **carray_setspace** (**carray** ∗, size_t, void ∗∗)
- void **carray_adjust** (**carray** ∗, void ∗∗)
- size_t **carray_getreadposition** (**carray** ∗)
- size_t **carray_readingsremaining** (**carray** ∗)
- void **carray_setreadposition** (**carray** ∗, int, void ∗∗)
- type ∗ **carray_getarray** (**carray** ∗)
- type **carray_read** (**carray** ∗, void ∗∗ok)
- void **carray_reverse** (**carray** ∗)
- **carray** ∗ **carray_reversed_TF** (**carray** ∗)
- void **carray_append** (**carray** ∗, **carray** ∗, void ∗∗)
- **carray** ∗ **carray_concat_TF** (**carray** ∗, **carray** ∗)
- char ∗ **carray_tostring_TF** (**carray** ∗, char ∗(type), char ∗, char ∗, char ∗, char ∗)
- hashtype **carray_hashcode** (**carray** ∗, hashtype(type))
- bool **carray_equal** (**carray** ∗, **carray** ∗, bool(type, type))
- int **carray_indexof** (**carray** ∗, type, bool(type, type))
- int **carray_lastindexof** (**carray** ∗, type, bool(type, type))
- bool **carray_contains** (**carray** ∗, type, bool(type, type))
- **carray** ∗ **carray_subcarray_TF** (**carray** ∗, int, int, void ∗∗)
- **carray** ∗ **carray_subcarraystep_TF** (**carray** ∗, int, int, int, void ∗∗)
- type ∗ **carray_subarray_TF** (**carray** ∗, int, int, void ∗∗)
- type ∗ **carray_subarraystep_TF** (**carray** ∗, int, int, int, void ∗∗)
- type ∗ **carray_toarray_TF** (**carray** ∗)
- type **carray_get** (**carray** ∗, int, void ∗∗)
- void **carray_add** (**carray** ∗, int, type, void ∗∗)
- void **carray_push** (**carray** ∗, type)
- void **carray_ins** (**carray** ∗, type)
- void **carray_set** (**carray** ∗, int, type, void ∗∗)
- void **carray_safeset** (**carray** ∗, int, type, void ∗∗)
- type **carray_remove** (**carray** ∗, int, void ∗∗)
- type **carray_pop** (**carray** ∗)
- bool **carray_remove_elt** (**carray** ∗, type, bool(type, type))

Variables

- char ∗ **Char_holder**
- signed char ∗ **sChar_holder**
- unsigned char ∗ **uChar_holder**
- short ∗ **Short_holder**
- signed short ∗ **sShort_holder**
- unsigned short ∗ **uShort_holder**
- int ∗ **Int_holder**
- signed int ∗ **sInt_holder**
- unsigned int ∗ **uInt_holder**
- long ∗ **Long_holder**
- long long ∗ **lLong_holder**
- signed long ∗ **sLong_holder**
- signed long long ∗ **slLong_holder**
- unsigned long ∗ **uLong_holder**
- unsigned long long ∗ **ulLong_holder**
- float ∗ **Float_holder**
- double ∗ **Double_holder**
- long double ∗ **lDouble_holder**
- bool ∗ **Bool_holder**
- char ∗ **String_holder**

7.2.1 Detailed Description

Header of the carray class. Contains all function declarations and preprocessor directives.

Index