

My library

Generated by Doxygen 1.9.1



<b>1 myLibrary homepage</b>	<b>1</b>
1.1 Hi!	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 File Documentation</b>	<b>5</b>
3.1 arrays.h File Reference	5
3.1.1 Detailed Description	6
3.1.2 Function Documentation	6
3.1.2.1 charBubbleSort()	6
3.1.2.2 chooseBubbleSort()	6
3.1.2.3 doubleBubbleSort()	7
3.1.2.4 floatBubbleSort()	7
3.1.2.5 intBubbleSort()	7
3.1.2.6 linearSearch()	7
3.1.2.7 printMatrix()	8
3.1.2.8 ptrBubbleSort()	9
3.1.2.9 quickSort()	9
3.2 constants.h File Reference	9
3.2.1 Detailed Description	10
3.2.2 Macro Definition Documentation	10
3.2.2.1 EQUAL	11
3.2.2.2 FALSE	11
3.2.2.3 GREATER	11
3.2.2.4 KEY_NOT_FOUND	11
3.2.2.5 NULL_POINTER_GIVEN	11
3.2.2.6 SMALLER	11
3.2.2.7 SUCCESS	12
3.2.2.8 TRUE	12
3.2.2.9 UNKNOWN_SPEC	12
3.2.2.10 UNSUPPORTED_ARCHITECTURE	12
3.3 macros.h File Reference	12
3.3.1 Detailed Description	13
3.3.2 Macro Definition Documentation	13
3.3.2.1 bubbleSort	14
3.3.2.2 cmp	14
3.4 myLibrary.h File Reference	15
3.4.1 Detailed Description	15
3.5 strings.h File Reference	16
3.5.1 Detailed Description	17
3.5.2 Function Documentation	17
3.5.2.1 changeLastCharacter()	17

---

3.5.2.2 copyOf() . . . . .	17
3.5.2.3 endsWith() . . . . .	18
3.5.2.4 getLength() . . . . .	18
3.5.2.5 getString() . . . . .	19
3.6 types.h File Reference . . . . .	19
3.6.1 Detailed Description . . . . .	19
3.6.2 Typedef Documentation . . . . .	20
3.6.2.1 byte . . . . .	20
3.6.2.2 spec_t . . . . .	20
3.6.2.3 string . . . . .	20
3.7 utility.h File Reference . . . . .	20
3.7.1 Detailed Description . . . . .	21
3.7.2 Function Documentation . . . . .	22
3.7.2.1 byteCmp() . . . . .	22
3.7.2.2 charCmp() . . . . .	22
3.7.2.3 chooseCmp() . . . . .	22
3.7.2.4 doubleCmp() . . . . .	23
3.7.2.5 falseIfTrue() . . . . .	23
3.7.2.6 floatCmp() . . . . .	23
3.7.2.7 getCmp() . . . . .	23
3.7.2.8 intCmp() . . . . .	24
3.7.2.9 ptrCmp() . . . . .	24
3.7.2.10 saferMalloc() . . . . .	24
3.7.2.11 saferRealloc() . . . . .	25
3.7.2.12 trueIfFalse() . . . . .	25
<b>Index</b>	<b>27</b>

# Chapter 1

## myLibrary homepage

### 1.1 Hi!

Actually I don't know what I should put here, so at the moment I just suggest you to go to the [files](#) section. The source code and binaries are available [here](#). [Here](#) there is a PDF version of the docs.



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">arrays.h</a>	Common tasks with arrays: sorting, searching, printing etc . . . . .	5
<a href="#">constants.h</a>	Definition of symbolic constants used by the library . . . . .	9
<a href="#">macros.h</a>	Macros for emulated overloading . . . . .	12
<a href="#">myLibrary.h</a>	Includes all other headers. Useful for rapid import . . . . .	15
<a href="#">strings.h</a>	Common tasks with strings . . . . .	16
<a href="#">types.h</a>	Collection of useful types . . . . .	19
<a href="#">utility.h</a>	Common tasks such as comparing variables, swap bools, allocate memory . . . . .	20





## Chapter 3

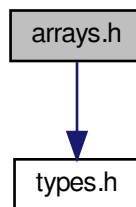
# File Documentation

### 3.1 arrays.h File Reference

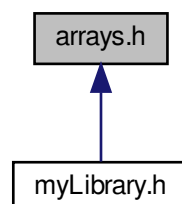
Common tasks with arrays: sorting, searching, printing etc.

```
#include "types.h"
```

Include dependency graph for arrays.h:



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



## Functions

- [byte chooseBubbleSort](#) (const [spec\\_t](#) spec, void \*arr, unsigned int size)  
*Bubble sort for arrays.*
- [byte quickSort](#) (const [spec\\_t](#) spec, void \*arr, int size)  
*Quick sort for arrays.*
- int [linearSearch](#) (const [spec\\_t](#) spec, const void \*arr, const void \*key, int size)  
*Linear search for arrays.*
- [byte printMatrix](#) (const [spec\\_t](#) spec, const void \*matrix, const unsigned int nRows, const unsigned int nColumns)  
*Print matrix of specified size with specified formatting.*
- [byte charBubbleSort](#) (char \*arr, unsigned int size)  
*Bubble sort for arrays of chars.*
- [byte intBubbleSort](#) (int \*arr, unsigned int size)  
*Bubble sort for arrays of ints.*
- [byte floatBubbleSort](#) (float \*arr, unsigned int size)  
*Bubble sort for arrays of floats.*
- [byte doubleBubbleSort](#) (double \*arr, unsigned int size)  
*Bubble sort for arrays of doubles.*
- [byte ptrBubbleSort](#) (int \*\*arr, unsigned int size)  
*Bubble sort for arrays of pointers.*

### 3.1.1 Detailed Description

Common tasks with arrays: sorting, searching, printing etc.

Author

Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

### 3.1.2 Function Documentation

#### 3.1.2.1 charBubbleSort()

```
byte charBubbleSort (
    char * arr,
    unsigned int size )
```

Bubble sort for arrays of chars.

Equivalent to `chooseBubbleSort ("%c", arr, size)`. Refer to [chooseBubbleSort\(\)](#)

#### 3.1.2.2 chooseBubbleSort()

```
byte chooseBubbleSort (
    const spec_t spec,
    void * arr,
    unsigned int size )
```

Bubble sort for arrays.

## Parameters

<i>spec</i>	Type specifier of the array to be sorted. Refer to <a href="#">spec_t</a> for supported types.
<i>arr</i>	Pointer to the first element of the array to be sorted
<i>size</i>	Number of elements of the array to be sorted

## Returns

The return code of the function

## Return values

<i>SUCCESS</i>	The array was correctly sorted
<i>UNKNOWN_SPEC</i>	Unknown id provided. The array has not been changed
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

### 3.1.2.3 doubleBubbleSort()

```
byte doubleBubbleSort (
    double * arr,
    unsigned int size )
```

Bubble sort for arrays of doubles.

Equivalent to `chooseBubbleSort("%lf", arr, size)`. Refer to [chooseBubbleSort\(\)](#)

### 3.1.2.4 floatBubbleSort()

```
byte floatBubbleSort (
    float * arr,
    unsigned int size )
```

Bubble sort for arrays of floats.

Equivalent to `chooseBubbleSort("%f", arr, size)`. Refer to [chooseBubbleSort\(\)](#)

### 3.1.2.5 intBubbleSort()

```
byte intBubbleSort (
    int * arr,
    unsigned int size )
```

Bubble sort for arrays of ints.

Equivalent to `chooseBubbleSort("%i", arr, size)`. Refer to [chooseBubbleSort\(\)](#)

### 3.1.2.6 linearSearch()

```
int linearSearch (
    const spec_t spec,
    const void * arr,
    const void * key,
    int size )
```

Linear search for arrays.

**Parameters**

<i>spec</i>	Type specifier of the array to be sorted. Refer to <a href="#">spec_t</a> for supported types
<i>arr</i>	Pointer to the first element of the array to be inspected
<i>key</i>	Pointer to the key
<i>size</i>	Number of elements of the array to be inspected

**Returns**

The index of the first occurrence of the key in the array or the return code of the function

**Return values**

<i>KEY_NOT_FOUND</i>	The key was not found
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

**3.1.2.7 printMatrix()**

```
byte printMatrix (
    const spec\_t spec,
    const void * matrix,
    const unsigned int nRows,
    const unsigned int nColumns )
```

Print matrix of specified size with specified formatting.

**Parameters**

<i>spec</i>	Type and format specifier used to print a cell. The printf() identifier formatting convention is supported. See <a href="#">spec_t</a> for details. Additional supported specifiers: "%hi" (numerical output for char)
-------------	--

**Note**

The format specifier must end with the letter of the type specifier. For example, "%5.3lf" is supported, "%5.3lf\n" or "%5.3lfTest" is not supported and nothing is printed

**Parameters**

<i>matrix</i>	Pointer to the first element of the matrix
<i>nRows</i>	Number of rows of the matrix
<i>nColumns</i>	Number of rows of the matrix

**Returns**

The return code of the function

## Return values

<i>SUCCESS</i>	The matrix was correctly printed
<i>UNKNOWN_SPEC</i>	Give type specifier was not recognised
<i>NULL_POINTER_GIVEN</i>	At least one among given pointer was NULL

## 3.1.2.8 ptrBubbleSort()

```
byte ptrBubbleSort (
    int ** arr,
    unsigned int size )
```

Bubble sort for arrays of pointers.

Equivalent to `chooseBubbleSort ("%p", arr, size)`. Refer to [chooseBubbleSort\(\)](#)

## 3.1.2.9 quickSort()

```
byte quickSort (
    const spec_t spec,
    void * arr,
    int size )
```

Quick sort for arrays.

## Parameters

<i>spec</i>	Type specifier of the array to be sorted. Refer to <a href="#">spec_t</a> for supported types
<i>arr</i>	Pointer to the first element of the array to be sorted
<i>size</i>	Number of elements of the array to be sorted

## Returns

The return code of the function

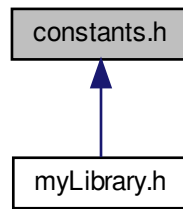
## Return values

<i>SUCCESS</i>	The array was correctly sorted
<i>UNKNOWN_SPEC</i>	Unknown id provided. The array has not been changed
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

## 3.2 constants.h File Reference

Definition of symbolic constants used by the library.

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



## Macros

- #define **GREATER** 1  
*Returned by typeCmp() functions when first argument is greater than the second.*
- #define **EQUAL** 0  
*Returned by typeCmp() functions when first argument is equal to the second.*
- #define **SMALLER** -1  
*Returned by typeCmp() functions when first argument is smaller than the second.*
- #define **UNSUPPORTED\_ARCHITECTURE** 64  
*Returned when pointers have unsupported size.*
- #define **TRUE** 0xFF  
*Bool value definition.*
- #define **FALSE** 0  
*Bool value definition.*
- #define **SUCCESS** 0  
*Returned when a function ended successfully.*
- #define **UNKNOWN\_SPEC** 101  
*Returned when an unknown specifier was provided.*
- #define **KEY\_NOT\_FOUND** -1  
*Returned by search algorithms when key was not found.*
- #define **NULL\_POINTER\_GIVEN** -64  
*Returned when a null pointer was given.*

### 3.2.1 Detailed Description

Definition of symbolic constants used by the library.

Author

Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

### 3.2.2 Macro Definition Documentation

### 3.2.2.1 EQUAL

```
#define EQUAL 0
```

Returned by *typeCmp()* functions when first argument is equal to the second.

### 3.2.2.2 FALSE

```
#define FALSE 0
```

Bool value definition.

### 3.2.2.3 GREATER

```
#define GREATER 1
```

Returned by *typeCmp()* functions when first argument is greater than the second.

### 3.2.2.4 KEY\_NOT\_FOUND

```
#define KEY_NOT_FOUND -1
```

Returned by search algorithms when key was not found.

### 3.2.2.5 NULL\_POINTER\_GIVEN

```
#define NULL_POINTER_GIVEN -64
```

Returned when a null pointer was given.

### 3.2.2.6 SMALLER

```
#define SMALLER -1
```

Returned by *typeCmp()* functions when first argument is smaller than the second.

### 3.2.2.7 SUCCESS

```
#define SUCCESS 0
```

Returned when a function ended successfully.

### 3.2.2.8 TRUE

```
#define TRUE 0xFF
```

Bool value definition.

### 3.2.2.9 UNKNOWN\_SPEC

```
#define UNKNOWN_SPEC 101
```

Returned when an unknown specifier was provided.

### 3.2.2.10 UNSUPPORTED\_ARCHITECTURE

```
#define UNSUPPORTED_ARCHITECTURE 64
```

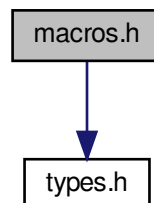
Returned when pointers have unsupported size.

## 3.3 macros.h File Reference

Macros for emulated overloading.

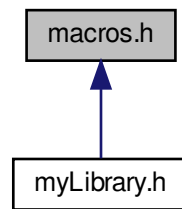
```
#include "types.h"
```

Include dependency graph for macros.h:





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



## Macros

- `#define cmp(a, b)`  
*Compare two values. Calls the right typeCmp() function.*
- `#define bubbleSort(arr, size)`  
*BubbleSort for arrays.*

### 3.3.1 Detailed Description

Macros for emulated overloading.

#### Author

Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

#### Note

Many of these macros work on C11 or newer compilers only. If they are not supported by your compiler you have to use the function the macro expands to in your case. For example, if you want to bubblesort an array of floats and the macro `bubbleSort()` is not supported by your compiler, you have to call `floatBubbleSort()` or `chooseBubbleSort()`

In some development environments, for example Vscode, calls to these macros can be reported as errors even if they are correct. If you use Vscode you have to set `"C_Cpp.default.cStandard": "c17"` in your `settings.json` file in order to avoid this error reportings

### 3.3.2 Macro Definition Documentation

### 3.3.2.1 bubbleSort

```
#define bubbleSort(
    arr,
    size )
```

#### Value:

```
_Generic(arr,
char *:    charBubbleSort,
int *:     intBubbleSort,
float *:   floatBubbleSort,
double *:  doubleBubbleSort,
void **:   ptrBubbleSort)(arr, size)
```

BubbleSort for arrays.

#### Returns

The return code of the function called

#### Parameters

<i>arr</i>	Pointer to the array to be sorted
<i>size</i>	Number of elements in the array to be sorted

#### Return values

<i>NULL_POINTER_GIVEN</i>	Pointer to the array to be sorted
<i>SUCCESS</i>	Array successfully sorted

### 3.3.2.2 cmp

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

#### Value:

```
_Generic((a, b),
char:    charCmp,
int:     intCmp,
float:   floatCmp,
double:  doubleCmp,
void *:  ptrCmp)(&a, &b)
```

Compare two values. Calls the right *typeCmp()* function.

#### Note

This macro must be called on variables. For example, `cmp(2, 3)` is not supported

#### Parameters

<i>a</i>	First value to be compared
<i>b</i>	Second value to be compared

### Returns

The return code of the function called

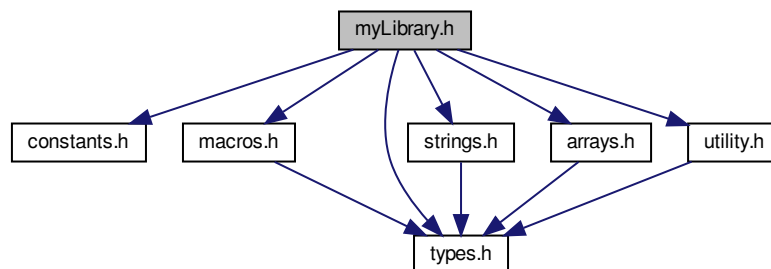
### Return values

<i>GREATER</i>	First element is greater than the second
<i>EQUAL</i>	First element is equal to the second
<i>SMALLER</i>	First element is smaller than the second

## 3.4 myLibrary.h File Reference

Includes all other headers. Useful for rapid import.

```
#include "constants.h"
#include "macros.h"
#include "types.h"
#include "strings.h"
#include "arrays.h"
#include "utility.h"
Include dependency graph for myLibrary.h:
```



### 3.4.1 Detailed Description

Includes all other headers. Useful for rapid import.

### Author

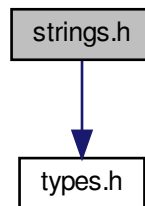
Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

## 3.5 strings.h File Reference

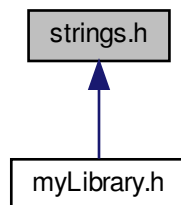
Common tasks with strings.

```
#include "types.h"
```

Include dependency graph for strings.h:



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



## Functions

- [string getString](#) ()  
*Reads from terminal a string of arbitrary length.*
- [byte endsWith](#) (const [string](#) str, const [string](#) suffix)  
*Check if a string ends with the specified substring.*
- [string changeLastCharacter](#) (const [string](#) str, char newCharacter)  
*Get string with different last character.*
- unsigned int [getLength](#) (const [string](#) str)  
*Get the lenght of a string.*
- [string copyOf](#) (const [string](#) src)  
*Get a copy of the given string.*

### 3.5.1 Detailed Description

Common tasks with strings.

Author

Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

### 3.5.2 Function Documentation

#### 3.5.2.1 `changeLastCharacter()`

```
string changeLastCharacter (
    const string str,
    char newCharacter )
```

Get string with different last character.

Parameters

<i>str</i>	The string you want to change the last character
<i>newCharacter</i>	The character you want to set as last character

Returns

A pointer to a string with the same characters of `str` and `newCharacter` as last character or a null pointer in case of errors

#### 3.5.2.2 `copyOf()`

```
string copyOf (
    const string src )
```

Get a copy of the given string.

Parameters

<i>src</i>	The string to be copied
------------	-------------------------

Returns

A pointer to the copy of the given string or a null pointer in case of errors

### 3.5.2.3 endsWith()

```
byte endsWith (
    const string str,
    const string suffix )
```

Check if a string ends with the specified substring.

#### Parameters

<i>str</i>	The string to be inspected
<i>suffix</i>	The string you want to check if <i>string</i> ends with

#### Returns

A boolean value

#### Return values

<i>TRUE</i>	<i>str</i> ends with <i>suffix</i>
<i>FALSE</i>	<i>str</i> does not end with <i>suffix</i>
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

### 3.5.2.4 getLength()

```
unsigned int getLength (
    const string str )
```

Get the length of a string.

#### Parameters

<i>str</i>	The string to be evaluated
------------	----------------------------

#### Returns

The length of the given string (terminator EXCLUDED) or the return code of the function

#### Return values

<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL
---------------------------	--

### 3.5.2.5 getString()

```
string getString ( )
```

Reads from terminal a string of arbitrary length.

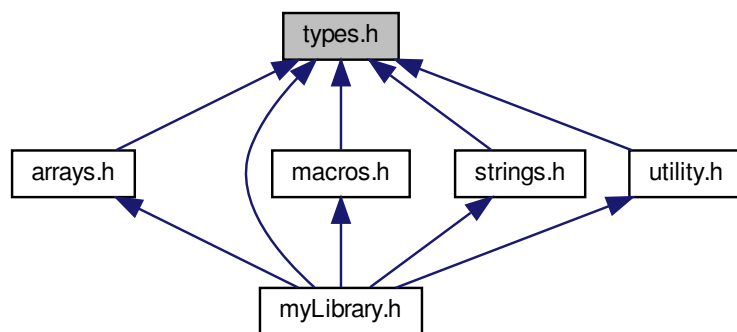
#### Returns

A char pointer to the first element of the string or a null pointer in case of errors

## 3.6 types.h File Reference

Collection of useful types.

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



### Typedefs

- typedef char [byte](#)  
*Alias for char, just to avoid confusion with 8 bit numbers and ASCII characters.*
- typedef char \* [spec\\_t](#)  
*Used to specify type of argument passed in functions that require a type specifier.*
- typedef char \* [string](#)  
*Alias for char \*, used when an array of char is actually used as a string.*

### 3.6.1 Detailed Description

Collection of useful types.

#### Author

Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

## 3.6.2 Typedef Documentation

### 3.6.2.1 byte

```
typedef char byte
```

Alias for char, just to avoid confusion with 8 bit numbers and ASCII characters.

### 3.6.2.2 spec\_t

```
typedef char* spec_t
```

Used to specify type of argument passed in functions that require a type specifier.

Supported specifiers: "%c" (char), "%i" (int), "%f" (float), "%lf" (double), "%p" (pointer)

#### Note

Some functions may not support some identifiers or may support additional identifiers. In those cases refer to that function documentation

### 3.6.2.3 string

```
typedef char* string
```

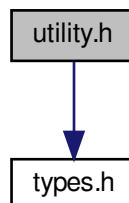
Alias for char \*, used when an array of char is actually used as a string.

## 3.7 utility.h File Reference

Common tasks such as comparing variables, swap bools, allocate memory.

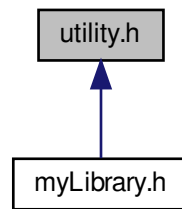
```
#include "types.h"
```

Include dependency graph for utility.h:





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



## Functions

- [byte chooseCmp](#) (const [spec\\_t](#) spec, const void \*a, const void \*b)  
*Compare two chars.*
- [byte charCmp](#) (const void \*a, const void \*b)  
*Compare two chars.*
- [byte byteCmp](#) (const void \*a, const void \*b)  
*Compare two bytes.*
- [byte intCmp](#) (const void \*a, const void \*b)  
*Compare two ints.*
- [byte floatCmp](#) (const void \*a, const void \*b)  
*Compare two floats.*
- [byte doubleCmp](#) (const void \*a, const void \*b)  
*Compare two doubles.*
- [byte ptrCmp](#) (const void \*a, const void \*b)  
*Compare two pointers.*
- void \* [getCmp](#) (const [spec\\_t](#) spec)  
*Choose comparison function based on given identifier.*
- [byte trueIfFalse](#) ([byte](#) \*value)  
*Set variable to `TRUE` if variable at provided address is 0.*
- [byte falseIfTrue](#) ([byte](#) \*value)  
*Set variable to `FALSE` if variable at provided address is not 0.*
- void \* [saferMalloc](#) (unsigned int bytes)  
*Return a pointer to a space in memory of specified size.*
- void \* [saferRealloc](#) (void \*pointer, unsigned int bytes)  
*Reallocate a space in memory.*

### 3.7.1 Detailed Description

Common tasks such as comparing variables, swap bools, allocate memory.

Author

Pietro Firpo ( [pietro.firpo@pm.me](mailto:pietro.firpo@pm.me) )

### 3.7.2 Function Documentation

#### 3.7.2.1 byteCmp()

```
byte byteCmp (
    const void * a,
    const void * b )
```

Compare two bytes.

Equivalent to `charCmp(a, b)`. Refer to [charCmp\(\)](#).

#### 3.7.2.2 charCmp()

```
byte charCmp (
    const void * a,
    const void * b )
```

Compare two chars.

Equivalent to `chooseCmp("%c", a, b)`. Refer to [chooseCmp\(\)](#)

#### 3.7.2.3 chooseCmp()

```
byte chooseCmp (
    const spec_t spec,
    const void * a,
    const void * b )
```

Compare two chars.

##### Parameters

<i>spec</i>	Type specifier of the values to be sorted. Refer to <a href="#">spec_t</a> for supported types.
<i>a</i>	Pointer to the first element to be compared
<i>b</i>	Pointer to the second element to be compared

##### Returns

Constant for the corresponding comparison result or the return code of the function

##### Return values

<i>GREATER</i>	First element is greater than the second
<i>EQUAL</i>	First element is equal to the second
<i>SMALLER</i>	First element is smaller than the second
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

### 3.7.2.4 doubleCmp()

```
byte doubleCmp (
    const void * a,
    const void * b )
```

Compare two doubles.

Equivalent to `chooseCmp ("%lf", a, b)`. Refer to [chooseCmp\(\)](#)

### 3.7.2.5 falseIfTrue()

```
byte falseIfTrue (
    byte * value )
```

Set variable to FALSE if variable at provided address is not 0.

#### Parameters

<i>value</i>	Pointer to the value to be evaluated
--------------	--------------------------------------

#### Returns

Return code of the function

#### Return values

<i>SUCCESS</i>	Function executed correctly
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

### 3.7.2.6 floatCmp()

```
byte floatCmp (
    const void * a,
    const void * b )
```

Compare two floats.

Equivalent to `chooseCmp ("%f", a, b)`. Refer to [chooseCmp\(\)](#)

### 3.7.2.7 getCmp()

```
void* getCmp (
    const spec_t spec )
```

Choose comparison function based on given identifier.

**Parameters**

<i>spec</i>	Specifier of the type of the data. Refer to <a href="#">spec_t</a>
-------------	--

**Returns**

Pointer to the right comparison function, `NULL` if identifier is not recognized or given pointer was `NULL`

**3.7.2.8 intCmp()**

```
byte intCmp (
    const void * a,
    const void * b )
```

Compare two ints.

Equivalent to `chooseCmp ("%i", a, b)`. Refer to [chooseCmp\(\)](#)

**3.7.2.9 ptrCmp()**

```
byte ptrCmp (
    const void * a,
    const void * b )
```

Compare two pointers.

Equivalent to `chooseCmp ("%p", a, b)`. Refer to [chooseCmp\(\)](#)

**3.7.2.10 saferMalloc()**

```
void* saferMalloc (
    unsigned int bytes )
```

Return a pointer to a space in memory of specified size.

Calls `malloc(bytes)` for a maximum of 10 times until it returns a not null pointer

**Parameters**

<i>bytes</i>	Number of bytes to allocate
--------------	-----------------------------

**Returns**

A pointer to the allocated memory or the return code of the function

## Return values

<i>NULL</i>	Could not allocate memory
-------------	---------------------------

**3.7.2.11 saferRealloc()**

```
void* saferRealloc (
    void * pointer,
    unsigned int bytes )
```

Reallocate a space in memory.

Calls `realloc(pointer, bytes)` for a maximum of 10 times until it returns a not null pointer

## Parameters

<i>pointer</i>	Pointer to the memory to be reallocated
<i>bytes</i>	Number of bytes to allocate

## Returns

A pointer to the allocated memory or the return code of the function

## Return values

<i>NULL</i>	Could not allocate memory
-------------	---------------------------

**3.7.2.12 trueIfFalse()**

```
byte trueIfFalse (
    byte * value )
```

Set variable to TRUE if variable at provided address is 0.

## Parameters

<i>value</i>	Pointer to the value to be evaluated
--------------	--------------------------------------

## Returns

Return code of the function

## Return values

<i>SUCCESS</i>	Function executed correctly
<i>NULL_POINTER_GIVEN</i>	At least one among given pointers was NULL

# Index

- arrays.h, [5](#)
  - charBubbleSort, [6](#)
  - chooseBubbleSort, [6](#)
  - doubleBubbleSort, [7](#)
  - floatBubbleSort, [7](#)
  - intBubbleSort, [7](#)
  - linearSearch, [7](#)
  - printMatrix, [8](#)
  - ptrBubbleSort, [9](#)
  - quickSort, [9](#)
- bubbleSort
  - macros.h, [13](#)
- byte
  - types.h, [20](#)
- byteCmp
  - utility.h, [22](#)
- changeLastCharacter
  - strings.h, [17](#)
- charBubbleSort
  - arrays.h, [6](#)
- charCmp
  - utility.h, [22](#)
- chooseBubbleSort
  - arrays.h, [6](#)
- chooseCmp
  - utility.h, [22](#)
- cmp
  - macros.h, [14](#)
- constants.h, [9](#)
  - EQUAL, [10](#)
  - FALSE, [11](#)
  - GREATER, [11](#)
  - KEY\_NOT\_FOUND, [11](#)
  - NULL\_POINTER\_GIVEN, [11](#)
  - SMALLER, [11](#)
  - SUCCESS, [11](#)
  - TRUE, [12](#)
  - UNKNOWN\_SPEC, [12](#)
  - UNSUPPORTED\_ARCHITECTURE, [12](#)
- copyOf
  - strings.h, [17](#)
- doubleBubbleSort
  - arrays.h, [7](#)
- doubleCmp
  - utility.h, [23](#)
- endsWith
  - strings.h, [17](#)
- EQUAL
  - constants.h, [10](#)
- FALSE
  - constants.h, [11](#)
- falseIfTrue
  - utility.h, [23](#)
- floatBubbleSort
  - arrays.h, [7](#)
- floatCmp
  - utility.h, [23](#)
- getCmp
  - utility.h, [23](#)
- getLength
  - strings.h, [18](#)
- getString
  - strings.h, [18](#)
- GREATER
  - constants.h, [11](#)
- intBubbleSort
  - arrays.h, [7](#)
- intCmp
  - utility.h, [24](#)
- KEY\_NOT\_FOUND
  - constants.h, [11](#)
- linearSearch
  - arrays.h, [7](#)
- macros.h, [12](#)
  - bubbleSort, [13](#)
  - cmp, [14](#)
- myLibrary.h, [15](#)
- NULL\_POINTER\_GIVEN
  - constants.h, [11](#)
- printMatrix
  - arrays.h, [8](#)
- ptrBubbleSort
  - arrays.h, [9](#)
- ptrCmp
  - utility.h, [24](#)
- quickSort
  - arrays.h, [9](#)
- saferMalloc

- utility.h, [24](#)
- saferRealloc
  - utility.h, [25](#)
- SMALLER
  - constants.h, [11](#)
- spec\_t
  - types.h, [20](#)
- string
  - types.h, [20](#)
- strings.h, [16](#)
  - changeLastCharacter, [17](#)
  - copyOf, [17](#)
  - endsWith, [17](#)
  - getLength, [18](#)
  - getString, [18](#)
- SUCCESS
  - constants.h, [11](#)
- TRUE
  - constants.h, [12](#)
- trueIfFalse
  - utility.h, [25](#)
- types.h, [19](#)
  - byte, [20](#)
  - spec\_t, [20](#)
  - string, [20](#)
- UNKNOWN\_SPEC
  - constants.h, [12](#)
- UNSUPPORTED\_ARCHITECTURE
  - constants.h, [12](#)
- utility.h, [20](#)
  - byteCmp, [22](#)
  - charCmp, [22](#)
  - chooseCmp, [22](#)
  - doubleCmp, [23](#)
  - falseIfTrue, [23](#)
  - floatCmp, [23](#)
  - getCmp, [23](#)
  - intCmp, [24](#)
  - ptrCmp, [24](#)
  - saferMalloc, [24](#)
  - saferRealloc, [25](#)
  - trueIfFalse, [25](#)