

My library

Generated by Doxygen 1.8.17

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 arrays.h File Reference	3
2.1.1 Detailed Description	4
2.1.2 Function Documentation	4
2.1.2.1 bubbleSort()	4
2.1.2.2 linearSearch()	5
2.1.2.3 printMatrix()	5
2.1.2.4 quickSort()	6
2.2 constants.h File Reference	6
2.2.1 Detailed Description	7
2.2.2 Macro Definition Documentation	7
2.2.2.1 EQUAL	8
2.2.2.2 FALSE	8
2.2.2.3 GREATER	8
2.2.2.4 KEY_NOT_FOUND	8
2.2.2.5 SMALLER	8
2.2.2.6 SUCCESS	8
2.2.2.7 TRUE	9
2.2.2.8 UNKNOWN_SPEC	9
2.2.2.9 UNSUPPORTED_ARCHITECTURE	9
2.3 myLibrary.h File Reference	9
2.3.1 Detailed Description	10
2.3.2 Macro Definition Documentation	10
2.3.2.1 NULL_POINTER_GIVEN	10
2.4 strings.h File Reference	10
2.4.1 Detailed Description	11
2.4.2 Function Documentation	11
2.4.2.1 changeLastCharacter()	11
2.4.2.2 copyOf()	12
2.4.2.3 endsWith()	12
2.4.2.4 getLength()	13
2.4.2.5 getString()	13
2.5 types.h File Reference	13
2.5.1 Detailed Description	14
2.5.2 Typedef Documentation	14
2.5.2.1 byte	14
2.5.2.2 spec_t	15
2.6 utility.h File Reference	15
2.6.1 Detailed Description	16

2.6.2 Function Documentation	16
2.6.2.1 byteCmp()	16
2.6.2.2 charCmp()	16
2.6.2.3 chooseCmp()	17
2.6.2.4 doubleCmp()	17
2.6.2.5 falseIfTrue()	18
2.6.2.6 floatCmp()	18
2.6.2.7 intCmp()	19
2.6.2.8 ptrCmp()	19
2.6.2.9 saferMalloc()	20
2.6.2.10 saferRealloc()	20
2.6.2.11 trueIfFalse()	22
 Index	 23

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

arrays.h	Common tasks with arrays: sorting	3
constants.h	Definition of symbolic constants used by the library	6
myLibrary.h	Includes all other headers. Useful for rapid import	9
strings.h	Common tasks with strings	10
types.h	Collection of useful types	13
utility.h	Common tasks such as comparing variables and swap bools when necessary	15

Chapter 2

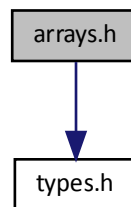
File Documentation

2.1 arrays.h File Reference

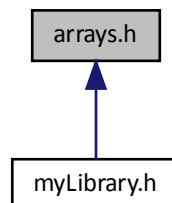
Common tasks with arrays: sorting.

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

Include dependency graph for arrays.h:



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



Functions

- `byte bubbleSort` (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.

2.1.1 Detailed Description

Common tasks with arrays: sorting.

Author

Pietro Firpo (pietro.firpo@pm.me)

2.1.2 Function Documentation

2.1.2.1 bubbleSort()

```
byte bubbleSort (
    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 spec_t 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

2.1.2.2 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 spec_t 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

2.1.2.3 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 spec_t 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

2.1.2.4 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 spec_t 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

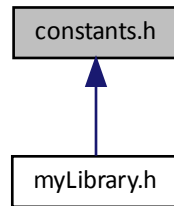
Return values

<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

2.2 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.

2.2.1 Detailed Description

Definition of symbolic constants used by the library.

Author

Pietro Firpo (pietro.firpo@pm.me)

2.2.2 Macro Definition Documentation

2.2.2.1 EQUAL

```
#define EQUAL 0
```

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

2.2.2.2 FALSE

```
#define FALSE 0
```

Bool value definition.

2.2.2.3 GREATER

```
#define GREATER 1
```

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

2.2.2.4 KEY_NOT_FOUND

```
#define KEY_NOT_FOUND -1
```

Returned by search algorithms when key was not found.

2.2.2.5 SMALLER

```
#define SMALLER -1
```

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

2.2.2.6 SUCCESS

```
#define SUCCESS 0
```

Returned when a function ended successfully.

2.2.2.7 TRUE

```
#define TRUE 0xFF
```

Bool value definition.

2.2.2.8 UNKNOWN_SPEC

```
#define UNKNOWN_SPEC 101
```

Returned when an unknown specifier was provided.

2.2.2.9 UNSUPPORTED_ARCHITECTURE

```
#define UNSUPPORTED_ARCHITECTURE 64
```

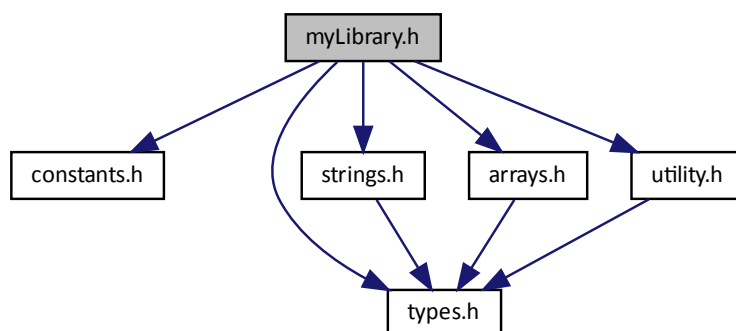
Returned when pointers have unsupported size.

2.3 myLibrary.h File Reference

Includes all other headers. Useful for rapid import.

```
#include "constants.h"  
#include "types.h"  
#include "strings.h"  
#include "arrays.h"  
#include "utility.h"
```

Include dependency graph for myLibrary.h:



Macros

- `#define NULL_POINTER_GIVEN -64`

2.3.1 Detailed Description

Includes all other headers. Useful for rapid import.

Author

Pietro Firpo (pietro.firpo@pm.me)

2.3.2 Macro Definition Documentation

2.3.2.1 NULL_POINTER_GIVEN

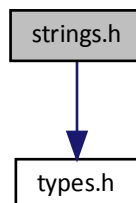
```
#define NULL_POINTER_GIVEN -64
```

2.4 strings.h File Reference

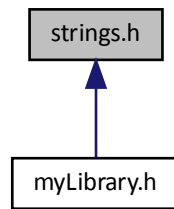
Common tasks with strings.

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

Include dependency graph for strings.h:



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



Functions

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

2.4.1 Detailed Description

Common tasks with strings.

Author

Pietro Firpo (pietro.firpo@pm.me)

2.4.2 Function Documentation

2.4.2.1 changeLastCharacter()

```
char* changeLastCharacter (  
    char * string,  
    char newCharacter )
```

Get string with different last character.

Parameters

<i>string</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 *string* and *newCharacter* as last character or a null pointer in case of errors

2.4.2.2 copyOf()

```
char* copyOf (
    const char * 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 or a null pointer in case of errors

2.4.2.3 endsWith()

```
byte endsWith (
    const char * string,
    const char * suffix )
```

Check if a string ends with the specified substring.

Parameters

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

2.4.2.4 getLength()

```
unsigned int getLength (
    const char * string )
```

Get the lenght of a string.

Parameters

<i>string</i>	pointer to the first element of the string to be evaluated
---------------	------------------------------------------------------------

Returns

The lenght of the given string or the error code of the function

Return values

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

2.4.2.5 getString()

```
char* getString ( )
```

Reads from terminal a string of arbitrary length.

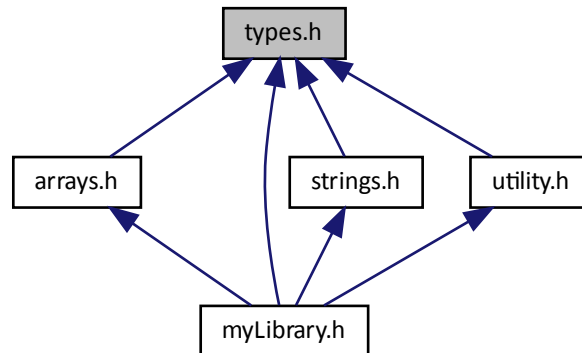
Returns

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

2.5 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.

2.5.1 Detailed Description

Collection of useful types.

Author

Pietro Firpo (pietro.firpo@pm.me)

2.5.2 Typedef Documentation

2.5.2.1 byte

```
typedef char byte
```

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

2.5.2.2 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

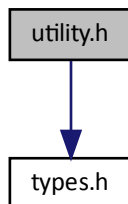
Some functions may not support some identifiers or may support additional identifiers. In those cases refer to that function documentation

2.6 utility.h File Reference

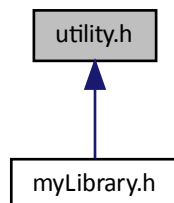
Common tasks such as comparing variables and swap bools when necessary.

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

Include dependency graph for utility.h:



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



Functions

- [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 * [chooseCmp](#) (const char *id)
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.

2.6.1 Detailed Description

Common tasks such as comparing variables and swap bools when necessary.

Author

Pietro Firpo (pietro.firpo@pm.me)

2.6.2 Function Documentation

2.6.2.1 byteCmp()

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

Compare two bytes.

The call is equivalent to `charCmp(a, b)`. Refer to [charCmp\(\)](#).

2.6.2.2 charCmp()

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

Compare two chars.

Parameters

<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

2.6.2.3 chooseCmp()

```
void* chooseCmp (
    const char * id )
```

Choose comparison function based on given identifier.

Parameters

<i>id</i>	Identifier of the type of the data. Refer to spec_t
-----------	---------------------------------------------------------------------

Returns

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

2.6.2.4 doubleCmp()

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

Compare two doubles.

Parameters

<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

2.6.2.5 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

2.6.2.6 floatCmp()

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

Compare two floats.

Parameters

<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

2.6.2.7 intCmp()

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

Compare two ints.

Parameters

<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

2.6.2.8 ptrCmp()

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

Compare two pointers.

Parameters

<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

2.6.2.9 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
-------------	---------------------------

2.6.2.10 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
-------------	---------------------------

2.6.2.11 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, [3](#)
 bubbleSort, [4](#)
 linearSearch, [5](#)
 printMatrix, [5](#)
 quickSort, [6](#)

bubbleSort
 arrays.h, [4](#)

byte
 types.h, [14](#)

byteCmp
 utility.h, [16](#)

changeLastCharacter
 strings.h, [11](#)

charCmp
 utility.h, [16](#)

chooseCmp
 utility.h, [17](#)

constants.h, [6](#)
 EQUAL, [7](#)
 FALSE, [8](#)
 GREATER, [8](#)
 KEY_NOT_FOUND, [8](#)
 SMALLER, [8](#)
 SUCCESS, [8](#)
 TRUE, [8](#)
 UNKNOWN_SPEC, [9](#)
 UNSUPPORTED_ARCHITECTURE, [9](#)

copyOf
 strings.h, [12](#)

doubleCmp
 utility.h, [17](#)

endsWith
 strings.h, [12](#)

EQUAL
 constants.h, [7](#)

FALSE
 constants.h, [8](#)

falseIfTrue
 utility.h, [18](#)

floatCmp
 utility.h, [18](#)

getLength
 strings.h, [13](#)

getString
 strings.h, [13](#)

GREATER
 constants.h, [8](#)

intCmp
 utility.h, [19](#)

KEY_NOT_FOUND
 constants.h, [8](#)

linearSearch
 arrays.h, [5](#)

myLibrary.h, [9](#)
 NULL_POINTER_GIVEN, [10](#)

NULL_POINTER_GIVEN
 myLibrary.h, [10](#)

printMatrix
 arrays.h, [5](#)

ptrCmp
 utility.h, [19](#)

quickSort
 arrays.h, [6](#)

saferMalloc
 utility.h, [20](#)

saferRealloc
 utility.h, [20](#)

SMALLER
 constants.h, [8](#)

spec_t
 types.h, [14](#)

strings.h, [10](#)
 changeLastCharacter, [11](#)
 copyOf, [12](#)
 endsWith, [12](#)
 getLength, [13](#)
 getString, [13](#)

SUCCESS
 constants.h, [8](#)

TRUE
 constants.h, [8](#)

trueIfFalse
 utility.h, [22](#)

types.h, [13](#)
 byte, [14](#)
 spec_t, [14](#)

UNKNOWN_SPEC

- constants.h, [9](#)
- UNSUPPORTED_ARCHITECTURE
 - constants.h, [9](#)
- utility.h, [15](#)
 - byteCmp, [16](#)
 - charCmp, [16](#)
 - chooseCmp, [17](#)
 - doubleCmp, [17](#)
 - falseIfTrue, [18](#)
 - floatCmp, [18](#)
 - intCmp, [19](#)
 - ptrCmp, [19](#)
 - saferMalloc, [20](#)
 - saferRealloc, [20](#)
 - trueIfFalse, [22](#)