

My library

Generated by Doxygen 1.8.17



<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 bubbleSort()	6
3.1.2.2 linearSearch()	7
3.1.2.3 printMatrix()	7
3.1.2.4 quickSort()	8
3.2 constants.h File Reference	8
3.2.1 Detailed Description	9
3.2.2 Macro Definition Documentation	9
3.2.2.1 EQUAL	10
3.2.2.2 FALSE	10
3.2.2.3 GREATER	10
3.2.2.4 KEY_NOT_FOUND	10
3.2.2.5 SMALLER	10
3.2.2.6 SUCCESS	10
3.2.2.7 TRUE	11
3.2.2.8 UNKNOWN_SPEC	11
3.2.2.9 UNSUPPORTED_ARCHITECTURE	11
3.3 myLibrary.h File Reference	11
3.3.1 Detailed Description	12
3.3.2 Macro Definition Documentation	12
3.3.2.1 NULL_POINTER_GIVEN	12
3.4 strings.h File Reference	12
3.4.1 Detailed Description	13
3.4.2 Function Documentation	13
3.4.2.1 changeLastCharacter()	13
3.4.2.2 copyOf()	14
3.4.2.3 endsWith()	14
3.4.2.4 getLength()	15
3.4.2.5 getString()	15
3.5 types.h File Reference	15
3.5.1 Detailed Description	16
3.5.2 Typedef Documentation	16
3.5.2.1 byte	16
3.5.2.2 spec_t	17

---

3.6 utility.h File Reference . . . . .	17
3.6.1 Detailed Description . . . . .	18
3.6.2 Macro Definition Documentation . . . . .	18
3.6.2.1 cmp . . . . .	19
3.6.3 Function Documentation . . . . .	19
3.6.3.1 byteCmp() . . . . .	19
3.6.3.2 charCmp() . . . . .	19
3.6.3.3 chooseCmp() . . . . .	19
3.6.3.4 doubleCmp() . . . . .	20
3.6.3.5 falseIfTrue() . . . . .	20
3.6.3.6 floatCmp() . . . . .	20
3.6.3.7 intCmp() . . . . .	21
3.6.3.8 ptrCmp() . . . . .	21
3.6.3.9 saferMalloc() . . . . .	21
3.6.3.10 saferRealloc() . . . . .	22
3.6.3.11 trueIfFalse() . . . . .	22
3.6.3.12 valCmp() . . . . .	23
<b>Index</b>	<b>25</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 is available [here](#)



## 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 . . . . .	5
<a href="#">constants.h</a>	Definition of symbolic constants used by the library . . . . .	8
<a href="#">myLibrary.h</a>	Includes all other headers. Useful for rapid import . . . . .	11
<a href="#">strings.h</a>	Common tasks with strings . . . . .	12
<a href="#">types.h</a>	Collection of useful types . . . . .	15
<a href="#">utility.h</a>	Common tasks such as comparing variables, swap bools, allocate memory . . . . .	17





## Chapter 3

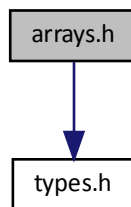
# File Documentation

### 3.1 arrays.h File Reference

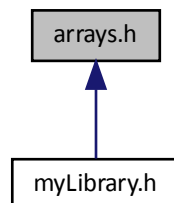
Common tasks with arrays: sorting.

```
#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.*

### 3.1.1 Detailed Description

Common tasks with arrays: sorting.

#### Author

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

### 3.1.2 Function Documentation

#### 3.1.2.1 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 <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

### 3.1.2.2 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 <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

### 3.1.2.3 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 <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

**3.1.2.4 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 <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

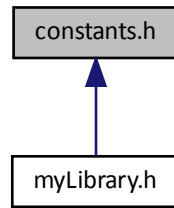
**Return values**

<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.*

### 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 SMALLER

```
#define SMALLER -1
```

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

### 3.2.2.6 SUCCESS

```
#define SUCCESS 0
```

Returned when a function ended successfully.

### 3.2.2.7 TRUE

```
#define TRUE 0xFF
```

Bool value definition.

### 3.2.2.8 UNKNOWN\_SPEC

```
#define UNKNOWN_SPEC 101
```

Returned when an unknown specifier was provided.

### 3.2.2.9 UNSUPPORTED\_ARCHITECTURE

```
#define UNSUPPORTED_ARCHITECTURE 64
```

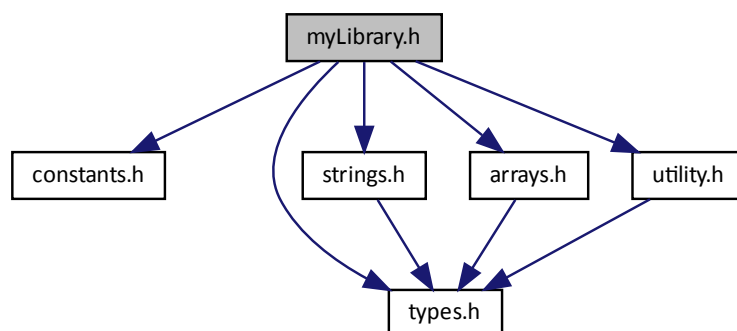
Returned when pointers have unsupported size.

## 3.3 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`

### 3.3.1 Detailed Description

Includes all other headers. Useful for rapid import.

#### Author

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

### 3.3.2 Macro Definition Documentation

#### 3.3.2.1 NULL\_POINTER\_GIVEN

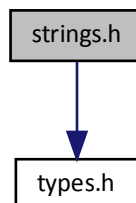
```
#define NULL_POINTER_GIVEN -64
```

## 3.4 strings.h File Reference

Common tasks with strings.

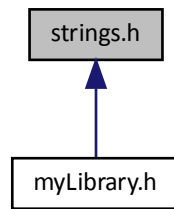
```
#include "types.h"
```

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.*

### 3.4.1 Detailed Description

Common tasks with strings.

Author

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

### 3.4.2 Function Documentation

#### 3.4.2.1 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

**3.4.2.2 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

**3.4.2.3 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

#### 3.4.2.4 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
---------------------------	--

#### 3.4.2.5 getString()

```
char* getString ( )
```

Reads from terminal a string of arbitrary length.

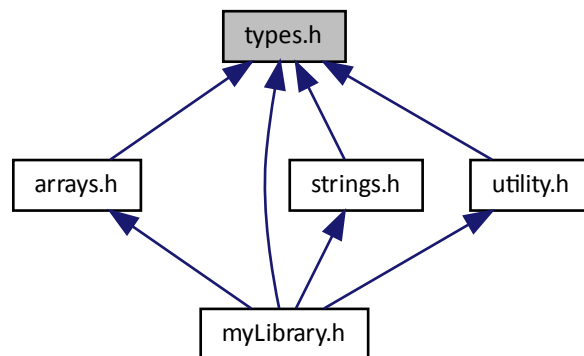
## Returns

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

## 3.5 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.*

### 3.5.1 Detailed Description

Collection of useful types.

Author

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

### 3.5.2 Typedef Documentation

#### 3.5.2.1 byte

```
typedef char byte
```

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

### 3.5.2.2 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

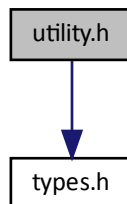
Some functions may not support some identifiers or may support additional identifiers. In those cases refer to that function documentation

## 3.6 utility.h File Reference

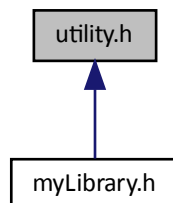
Common tasks such as comparing variables, swap bools, allocate memory.

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

Include dependency graph for utility.h:



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



## Macros

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

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

## Functions

- `byte valCmp (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 * chooseCmp (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.*

### 3.6.1 Detailed Description

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

#### Author

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

### 3.6.2 Macro Definition Documentation

### 3.6.2.1 cmp

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

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

#### Note

Works only on C11 or newer compilers

#### Returns

The return code of the function code

#### Return values

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

## 3.6.3 Function Documentation

### 3.6.3.1 byteCmp()

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

Compare two bytes.

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

### 3.6.3.2 charCmp()

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

Compare two chars.

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

### 3.6.3.3 chooseCmp()

```
void* chooseCmp (  
    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`

**3.6.3.4 doubleCmp()**

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

Compare two doubles.

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

**3.6.3.5 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 <code>NULL</code>

**3.6.3.6 floatCmp()**

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



Compare two floats.

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

### 3.6.3.7 intCmp()

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

Compare two ints.

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

### 3.6.3.8 ptrCmp()

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

Compare two pointers.

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

### 3.6.3.9 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
-------------	---------------------------

### 3.6.3.10 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
-------------	---------------------------

### 3.6.3.11 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 <code>NULL</code>

### 3.6.3.12 valCmp()

```
byte valCmp (
    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



# Index

- arrays.h, [5](#)
  - bubbleSort, [6](#)
  - linearSearch, [7](#)
  - printMatrix, [7](#)
  - quickSort, [8](#)
- bubbleSort
  - arrays.h, [6](#)
- byte
  - types.h, [16](#)
- byteCmp
  - utility.h, [19](#)
- changeLastCharacter
  - strings.h, [13](#)
- charCmp
  - utility.h, [19](#)
- chooseCmp
  - utility.h, [19](#)
- cmp
  - utility.h, [18](#)
- constants.h, [8](#)
  - EQUAL, [9](#)
  - FALSE, [10](#)
  - GREATER, [10](#)
  - KEY\_NOT\_FOUND, [10](#)
  - SMALLER, [10](#)
  - SUCCESS, [10](#)
  - TRUE, [10](#)
  - UNKNOWN\_SPEC, [11](#)
  - UNSUPPORTED\_ARCHITECTURE, [11](#)
- copyOf
  - strings.h, [14](#)
- doubleCmp
  - utility.h, [20](#)
- endsWith
  - strings.h, [14](#)
- EQUAL
  - constants.h, [9](#)
- FALSE
  - constants.h, [10](#)
- falseIfTrue
  - utility.h, [20](#)
- floatCmp
  - utility.h, [20](#)
- getLength
  - strings.h, [15](#)
- getString
  - strings.h, [15](#)
- GREATER
  - constants.h, [10](#)
- intCmp
  - utility.h, [21](#)
- KEY\_NOT\_FOUND
  - constants.h, [10](#)
- linearSearch
  - arrays.h, [7](#)
- myLibrary.h, [11](#)
  - NULL\_POINTER\_GIVEN, [12](#)
- NULL\_POINTER\_GIVEN
  - myLibrary.h, [12](#)
- printMatrix
  - arrays.h, [7](#)
- ptrCmp
  - utility.h, [21](#)
- quickSort
  - arrays.h, [8](#)
- saferMalloc
  - utility.h, [21](#)
- saferRealloc
  - utility.h, [21](#)
- SMALLER
  - constants.h, [10](#)
- spec\_t
  - types.h, [16](#)
- strings.h, [12](#)
  - changeLastCharacter, [13](#)
  - copyOf, [14](#)
  - endsWith, [14](#)
  - getLength, [15](#)
  - getString, [15](#)
- SUCCESS
  - constants.h, [10](#)
- TRUE
  - constants.h, [10](#)
- trueIfFalse
  - utility.h, [22](#)
- types.h, [15](#)
  - byte, [16](#)

spec\_t, [16](#)

UNKNOWN\_SPEC  
constants.h, [11](#)

UNSUPPORTED\_ARCHITECTURE  
constants.h, [11](#)

utility.h, [17](#)  
byteCmp, [19](#)  
charCmp, [19](#)  
chooseCmp, [19](#)  
cmp, [18](#)  
doubleCmp, [20](#)  
falseIfTrue, [20](#)  
floatCmp, [20](#)  
intCmp, [21](#)  
ptrCmp, [21](#)  
saferMalloc, [21](#)  
saferRealloc, [21](#)  
trueIfFalse, [22](#)  
valCmp, [22](#)

valCmp  
utility.h, [22](#)