My library

Generated by Doxygen 1.9.1

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:

Chapter 3

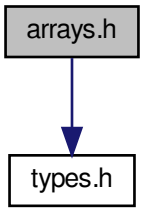
File Documentation

3.1 arrays.h File Reference

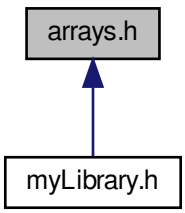
Common tasks with arrays: sorting, searching, printing etc.

```
#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)

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

spec	Type specifier of the array to be sorted. Refer to spec_t for supported types.
arr	Pointer to the first element of the array to be sorted
size	Number of elements of the array to be sorted

Returns

The return code of the function

Return values

SUCCESS	The array was correctly sorted
UNKNOWN_SPEC	Unknown id provided. The array has not been changed
NULL_POINTER_GIVEN	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

spec	Type specifier of the array to be sorted. Refer to spec_t for supported types
arr	Pointer to the first element of the array to be inspected
key	Pointer to the key
size	Number of elements of the array to be inspected

Returns

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

Return values

KEY_NOT_FOUND	The key was not found
NULL_POINTER_GIVEN	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

spec	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

matrix	Pointer to the first element of the matrix
nRows	Number of rows of the matrix
nColumns	Number of rows of the matrix

Returns

The return code of the function

Return values

SUCCESS	The matrix was correctly printed
UNKNOWN_SPEC	Give type specifier was not recognised
NULL_POINTER_GIVEN	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

spec	Type specifier of the array to be sorted. Refer to spec_t for supported types
arr	Pointer to the first element of the array to be sorted
size	Number of elements of the array to be sorted

Returns

The return code of the function

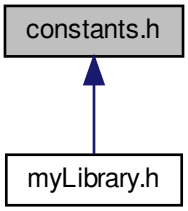
Return values

SUCCESS	The array was correctly sorted
UNKNOWN_SPEC	Unknown id provided. The array has not been changed
NULL_POINTER_GIVEN	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 grater 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)

3.2.2 Macro Definition Documentation

3.2.2.1 EQUAL

```
#define EQUAL 0
```

Returned by *type*Cmp() 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 *type*Cmp() functions when first argument is grater 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 *type*Cmp() 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
```

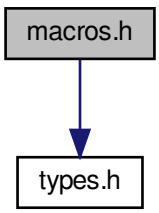
Returned when pointers have unsupported size.

3.3 macros.h File Reference

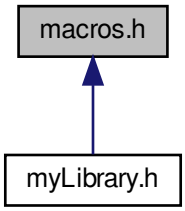
Macros for emulated overloading.

```
#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`)

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

arr	Pointer to the array to be sorted
size	Number of elements in the array to be sorted

Return values

NULL_POINTER_GIVEN	Pointer to the array to be sorted
SUCCESS	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 *type*Cmp() function.

Note

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

Parameters

a	First value to be compared
b	Second value to be compared

Returns

The return code of the function called

Return values

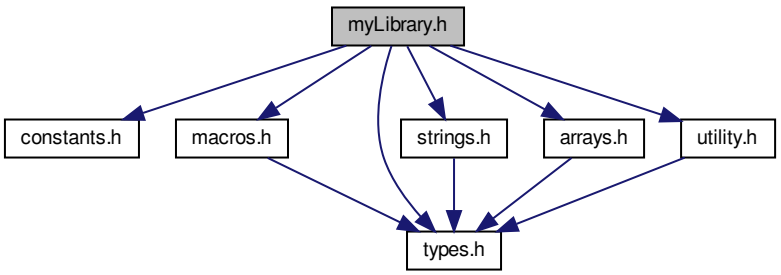
GREATER	First element is grater than the second
EQUAL	First element is equal to the second
SMALLER	First element is smaller than the second

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

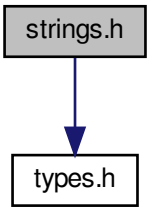
Pietro Firpo (pietro.firpo@pm.me)

3.5 strings.h File Reference

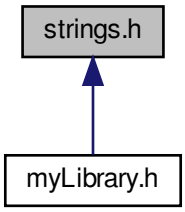
Common tasks with strings.

```
#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)

3.5.2 Function Documentation

3.5.2.1 changeLastCharacter()

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

Get string with different last character.

Parameters

str	The string you want to change the last character
newCharacter	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

src	The string to be copied

Returns

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

3.5.2.3 endsWith()

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

Check if a string ends with the specified substring.

Parameters

str	The string to be inspected
suffix	The string you want to check if `string`ends with

Returns

A boolean value

Return values

TRUE	`str` ends with `suffix`
FALSE	`str` does not end with `suffix`
NULL_POINTER_GIVEN	At least one among given pointers was NULL

3.5.2.4 getLength()

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

Get the lenght of a string.

Parameters

str	The string to be evaluated

Returns

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

Return values

NULL_POINTER_GIVEN	At least one among given pointers was NULL

3.5.2.5 getString()

```
string getString ( )
```

Reads from terminal a string of arbitrary length.

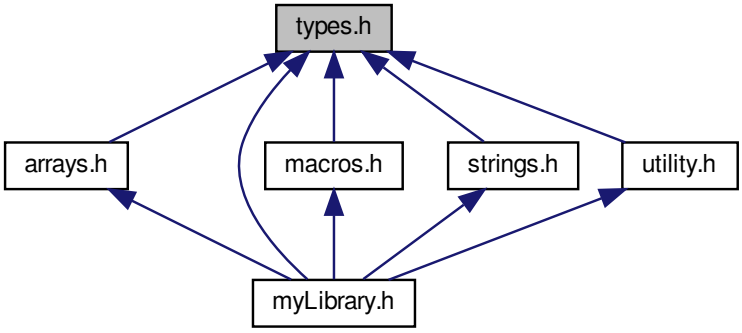
Returns

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)

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
```

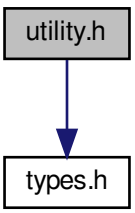
Alias for char *, used when an array of char is actually used as a string.

3.7 utility.h File Reference

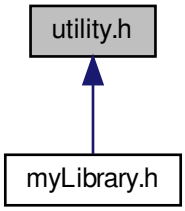
Common tasks such as comparing variables, swap bools, allocate memory.

```
#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)

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

spec	Type specifier of the values to be sorted. Refer to spec_t for supported types.
a	Pointer to the first element to be compared
b	Pointer to the second element to be compared

Returns

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

Return values

GREATER	First element is grater than the second
EQUAL	First element is equal to the second
SMALLER	First element is smaller than the second
NULL_POINTER_GIVEN	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

value	Pointer to the value to be evaluated

Returns

Return code of the function

Return values

SUCCESS	Function executed correctly
NULL_POINTER_GIVEN	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

spec	Specifier 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

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

bytes	Number of bytes to allocate

Returns

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

Return values

NULL	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

pointer	Pointer to the memory to be reallocated
bytes	Number of bytes to allocate

Returns

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

Return values

NULL	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

value	Pointer to the value to be evaluated

Returns

Return code of the function

Return values

SUCCESS	Function executed correctly
NULL_POINTER_GIVEN	At least one among given pointers was NULL

Index