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

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 ArrayList Struct Reference

`#include <types.h>`

Data Fields

- spec_t type
- void * body
- unsigned int size

4.1.1 Field Documentation

4.1.1.1 body

`void* ArrayList::body`

4.1.1.2 size

`unsigned int ArrayList::size`

4.1.1.3 type

`spec_t ArrayList::type`

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

- types.h

Chapter 5

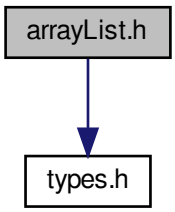
File Documentation

5.1 arrayList.h File Reference

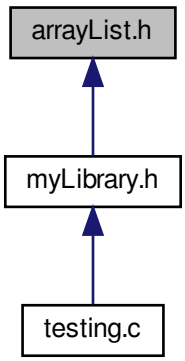
Functions for working with dynArrays.

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

Include dependency graph for arrayList.h:



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



Macros

- #define appendToAL(arr, element) _Generic(element, char: appendCharToAL)(arr, element)

  Append an element to an ArrayList.
- #define newALFromArray(arr, size) _Generic(arr, char∗: newALFromCharArray)(arr, size)

  Create an ArrayList from a static array.
- #define insertToAL(arr, element, index) _Generic(element, char: insertCharToAL)(arr, element, index)

  insert element into ArrayList
- #define setALElement(arr, element, index) _Generic(element, char: setALChar)(arr, element, index)

  Set an element of an ArrayList.

Functions

- ArrayList newAL ()

  Allocate a new ArrayList.
- ArrayList newALFromAL (const ArrayList arr)

  Get a copy of an ArrayList.
- void mergeAL (ArrayList arr1, const ArrayList arr2)

  Merge two ArrayList.
- void sliceAL (ArrayList arr, unsigned int begin, unsigned int end)

  Slice an ArrayList.
- void printAL (const spec_t spec, const ArrayList arr)

  Print an ArrayList content.
- void removeFromAL (ArrayList arr, unsigned int index)

  Remove an element from an ArrayList.
- void getFromAL (const ArrayList arr, unsigned int index, void ∗dest)

  Get an element from an ArrayList.
- void deleteAL (ArrayList arr)

  Delete an ArrayList.
- byte areALEqual (ArrayList arr1, ArrayList arr2)

  Compare two ArrayList.
- void reverseAL (ArrayList arr)

  Reverse an ArrayList.
- ArrayList newALFromCharArray (const char arr[ ], unsigned int size)

  Create an ArrayList from an array of chars.
- ArrayList newALFromByteArray (const char arr[ ], unsigned int size)

  Alias for newALFromCharArray(). Used to create ArrayList from byte array. Refer to newALFromCharArray()
- void appendCharToAL (ArrayList arr, char element)

  Insert a char at the end of an ArrayList.
- void insertCharToAL (ArrayList arr, char element, unsigned int index)

  Insert a char at a specified position of an ArrayList.
- void setALChar (ArrayList arr, char element, unsigned int index)

  Set value of an element of an ArrayList.

5.1.1 Detailed Description

Functions for working with dynArrays.

Author

Pietro Firpo (pietro.firpo@pm.me)

5.1.2 Macro Definition Documentation

5.1.2.1 appendToAL

```
#define appendToAL(
              arr,
              element ) _Generic(element, char:  appendCharToAL)(arr, element)
```

Append an element to an ArrayList.

Parameters

arr	The ArrayList you want to append an item to
element	The element you want to append to `arr`

Note

When you want to append something hardcoded that is not an `int` but the compiler treats as an `int` by default (a `byte`, a `char`, an integer `float` or `double`), you must specify a casting to that type. For example, if you want to append the number `42` to an ArrayList named `arr`, you must use `appendToAL(arr, (byte)42)`

5.1.2.2 insertToAL

```
#define insertToAL(
              arr,
              element,
              index ) _Generic(element, char:  insertCharToAL)(arr, element, index)
```

insert element into ArrayList

Parameters

arr	The ArrayList you want to insert an element into
element	The element you want to insert into `arr`
index	The index you want to insert `element` to

Note

When you want to insert something hardcoded that is not an `int` but the compiler treats as an `int` by default (a `byte`, a `char`, an integer `float` or `double`), you must specify a casting to that type. For example, if you want to insert the number `42` at index `4` into an ArrayList named `arr`, you must use `insertToAl(arr, (byte)42, 4)`

5.1.2.3 newALFromArray

```
#define newALFromArray(
              arr,
              size ) _Generic(arr, char*:  newALFromCharArray)(arr, size)
```

Create an ArrayList from a static array.

Parameters

arr	The array you want to create an ArrayList from
size	The size of `arr`

Returns

An ArrayList containing all the elements of `arr`void

5.1.2.4 setALElement

```
#define setALElement(
              arr,
              element,
              index ) _Generic(element, char:  setALChar)(arr, element, index)
```

Set an element of an ArrayList.

Parameters

arr	The :ArrayList you want to change an element
element	The element you want to set an item of `arr` to
index	The index of the element of the ArrayList you want to set to `element`

5.1.3 Function Documentation

5.1.3.1 appendCharToAL()

```
void appendCharToAL (
             ArrayList arr,
             char element )
```

Insert a char at the end of an ArrayList.

Parameters

arr	The ArrayList you want to append a char to
element	The char you want to append to `arr`

5.1.3.2 areALEqual()

```
byte areALEqual (
            ArrayList arr1,
            ArrayList arr2 )
```

Compare two ArrayList.

Parameters

arr1	The first ArrayList you want to compare
arr2	The second ArrayList you want to compare

Returns

The result of the comparison

Return values

TRUE	`arr1` and `arr2` have equal type, equal length and equal contents
FALSE	`arr1` and `arr2` do not have equal type, equal length or equal contents

5.1.3.3 deleteAL()

```
void deleteAL (
            ArrayList arr )
```

Delete an ArrayList.

Parameters

arr	The ArrayList you want to delete

5.1.3.4 getFromAL()

```
void getFromAL (
```

```
            const ArrayList arr,
            unsigned int index,
            void * dest )
```

Get an element from an ArrayList.

Parameters

arr	The ArrayList you want to get the item from
index	The index of the element you want to get
dest	The address of the variable you want to store the result in

5.1.3.5 insertCharToAL()

```
void insertCharToAL (
            ArrayList arr,
            char element,
            unsigned int index )
```

Insert a char at a specified position of an ArrayList.

Parameters

arr	The ArrayList you want to insert the char into
element	The char you want to insert into `arr`
index	The position you want to insert `element` at

5.1.3.6 mergeAL()

```
void mergeAL (
            ArrayList arr1,
            const ArrayList arr2 )
```

Merge two ArrayList.

Parameters

arr1	The first ArrayList to be merged, where the merged ArrayList is saved
arr2	The second ArrayList to be merged

5.1.3.7 newAL()

```
ArrayList newAL ( )
```

Allocate a new ArrayList.

Returns

An empty ArrayList

5.1.3.8 newALFromAL()

```
ArrayList newALFromAL (
            const ArrayList arr )
```

Get a copy of an ArrayList.

Parameters

arr	The ArrayList you want to copy

Returns

A copy of `arr`

5.1.3.9 newALFromByteArray()

```
ArrayList newALFromByteArray (
            const char arr[],
            unsigned int size )
```

Alias for newALFromCharArray(). Used to create ArrayList from byte array. Refer to newALFromCharArray()

5.1.3.10 newALFromCharArray()

```
ArrayList newALFromCharArray (
            const char arr[],
            unsigned int size )
```

Create an ArrayList from an array of chars.

Parameters

arr	The array you want to create the ArrayList from
size	The size of `arr`

Returns

An ArrayList of type char containing the elements in `arr` in the same order

5.1.3.11 printAL()

```
void printAL (
            const spec_t spec,
            const ArrayList arr )
```

Print an ArrayList content.

Parameters

spec	The type and format specifier you want to use to print the single element of the ArrayList
arr	The ArrayList you want to print

5.1.3.12 removeFromAL()

```
void removeFromAL (
            ArrayList arr,
            unsigned int index )
```

Remove an element from an ArrayList.

Parameters

arr	The ArrayList you want to delete an element from
index	The index of the element you want to delete

5.1.3.13 reverseAL()

```
void reverseAL (
            ArrayList arr )
```

Reverse an ArrayList.

Parameters

arr	The array you want to reverse

5.1.3.14 setALChar()

```
void setALChar (
            ArrayList arr,
            char element,
            unsigned int index )
```

Set value of an element of an ArrayList.

Parameters

arr	The ArrayList you want to edit
element	The element you want to set
index	The index of the element you want to change

5.1.3.15 sliceAL()

```
void sliceAL (
            ArrayList arr,
            unsigned int begin,
            unsigned int end )
```

Slice an ArrayList.

Parameters

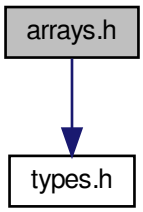
arr	The ArrayList you want to slice, where the sliced ArrayList is saved
begin	The index of the beginning of the slice
end	The index of the end of the slice

5.2 arrays.h File Reference

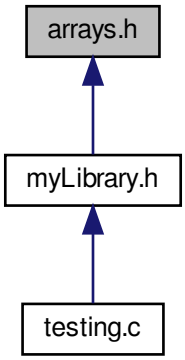
Common tasks with arrays: sorting, searching, printing etc.

```
#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 chooseQuickSort (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 n↩ Columns)

  Print matrix of specified size with specified formatting.
- byte charBubbleSort (char ∗arr, unsigned int size)

  Bubblesort for arrays of chars.
- byte intBubbleSort (int ∗arr, unsigned int size)

Bubblesort for arrays of ints.

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

  Bubblesort for arrays of floats.
- byte doubleBubbleSort (double *arr, unsigned int size)

  Bubblesort for arrays of doubles.
- byte ptrBubbleSort (void **arr, unsigned int size)

  Bubblesort for arrays of pointers.
- byte charQuickSort (char *arr, unsigned int size)

  Quicksort for arrays of chars.
- byte intQuickSort (int *arr, unsigned int size)

  Quicksort for arrays of ints.
- byte floatQuickSort (float *arr, unsigned int size)

  Quicksort for arrays of floats.
- byte doubleQuickSort (double *arr, unsigned int size)

  Quicksort for arrays of doubles.
- byte ptrQuickSort (void **arr, unsigned int size)

  Quicksort for arrays of pointers.

5.2.1 Detailed Description

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

Author

Pietro Firpo (pietro.firpo@pm.me)

5.2.2 Function Documentation

5.2.2.1 charBubbleSort()

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

Bubblesort for arrays of chars.

Equivalent to `chooseBubbleSort("%c", arr, size)`. Refer to chooseBubbleSort()

5.2.2.2 charQuickSort()

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

Quicksort for arrays of chars.

Equivalent to `chooseQuickSort("%c", arr, size)`. Refer to chooseQuickSort()

5.2.2.3 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	At least one among given pointers was null

5.2.2.4 chooseQuickSort()

```
byte chooseQuickSort (
            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	At least one among given pointers was null

5.2.2.5 doubleBubbleSort()

```
byte doubleBubbleSort (
```

```
            double * arr,
            unsigned int size )
```

Bubblesort for arrays of doubles.

Equivalent to `chooseBubbleSort("%lf", arr, size)`. Refer to chooseBubbleSort()

5.2.2.6 doubleQuickSort()

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

Quicksort for arrays of doubles.

Equivalent to `chooseQuickSort("%lf", arr, size)`. Refer to chooseQuickSort()

5.2.2.7 floatBubbleSort()

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

Bubblesort for arrays of floats.

Equivalent to `chooseBubbleSort("%f", arr, size)`. Refer to chooseBubbleSort()

5.2.2.8 floatQuickSort()

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

Quicksort for arrays of floats.

Equivalent to `chooseQuickSort("%f", arr, size)`. Refer to chooseQuickSort()

5.2.2.9 intBubbleSort()

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

Bubblesort for arrays of ints.

Equivalent to `chooseBubbleSort("%i", arr, size)`. Refer to chooseBubbleSort()

5.2.2.10 intQuickSort()

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

Quicksort for arrays of ints.

Equivalent to `chooseQuickSort("%i", arr, size)`. Refer to chooseQuickSort()

5.2.2.11 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	At least one among given pointers was null

5.2.2.12 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	At least one among given pointer was null

5.2.2.13 ptrBubbleSort()

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

Bubblesort for arrays of pointers.

Equivalent to `chooseBubbleSort("%p", arr, size)`. Refer to chooseBubbleSort()

5.2.2.14 ptrQuickSort()

```
byte ptrQuickSort (
            void ** arr,
            unsigned int size )
```

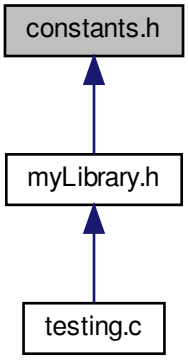
Quicksort for arrays of pointers.

Equivalent to `chooseQuickSort("%p", arr, size)`. Refer to chooseQuickSort()

5.3 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 27

  Returned when a function of the library ended successfully.
- #define UNKNOWN_SPEC 101

  Returned when an unknown specifier was provided.
- #define KEY_NOT_FOUND -1

  Returned by search functions of the library when key was not found.
- #define NULL_POINTER -64

  Returned when a null pointer was given.
- #define ALLOC_ERROR 37

  Returned when a function of the library could not allocate memory.
- #define DIFFERENT_TYPES -13

  Returned when given dynArrays have different types.
- #define UNDEFINED_TYPES -3

  Returned when both ::Arraylist elements have no defined type.

5.3.1 Detailed Description

Definition of symbolic constants used by the library.

Author

Pietro Firpo (pietro.firpo@pm.me)

5.3.2 Macro Definition Documentation

5.3.2.1 ALLOC_ERROR

```
#define ALLOC_ERROR 37
```

Returned when a function of the library could not allocate memory.

5.3.2.2 DIFFERENT_TYPES

```
#define DIFFERENT_TYPES -13
```

Returned when given dynArrays have different types.

5.3.2.3 EQUAL

```
#define EQUAL 0
```

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

5.3.2.4 FALSE

```
#define FALSE 0
```

Bool value definition.

5.3.2.5 GREATER

```
#define GREATER 1
```

Returned by *type*Cmp() functions when first argument is grater than the second.

5.3.2.6 KEY_NOT_FOUND

```
#define KEY_NOT_FOUND -1
```

Returned by search functions of the library when key was not found.

5.3.2.7 NULL_POINTER

```
#define NULL_POINTER -64
```

Returned when a null pointer was given.

5.3.2.8 SMALLER

```
#define SMALLER -1
```

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

5.3.2.9 SUCCESS

```
#define SUCCESS 27
```

Returned when a function of the library ended successfully.

5.3.2.10 TRUE

```
#define TRUE 0xFF
```

Bool value definition.

5.3.2.11 UNDEFINED_TYPES

```
#define UNDEFINED_TYPES -3
```

Returned when both ::Arraylist elements have no defined type.

5.3.2.12 UNKNOWN_SPEC

```
#define UNKNOWN_SPEC 101
```

Returned when an unknown specifier was provided.

5.3.2.13 UNSUPPORTED_ARCHITECTURE

```
#define UNSUPPORTED_ARCHITECTURE 64
```

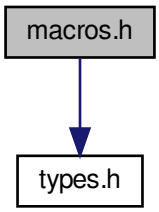
Returned when pointers have unsupported size.

5.4 macros.h File Reference

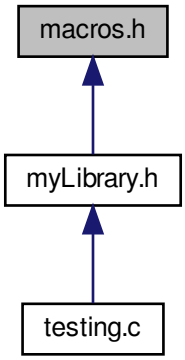
Macros for emulated overloading.

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

  Compare two values. Calls the right typeCmp() function.
- #define bubbleSort(arr, size) _Generic(arr, char ∗: charBubbleSort, int ∗: intBubbleSort, float ∗↩ :floatBubbleSort, double ∗: doubleBubbleSort, void ∗∗: ptrBubbleSort)(arr, size)

  BubbleSort for arrays.
- #define quickSort(arr, size) _Generic(arr, char ∗: charQuickSort, int ∗: intQuickSort, float ∗:floatQuickSort, double ∗: doubleQuickSort, void ∗∗: ptrQuickSort)(arr, size)

  Quicksort for arrays.

5.4.1 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

5.4.2 Macro Definition Documentation

5.4.2.1 bubbleSort

```
#define bubbleSort(
              arr,
              size ) _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	Pointer to the array to be sorted
SUCCESS	Array successfully sorted

5.4.2.2 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 *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

5.4.2.3 quickSort

```
#define quickSort(
              arr,
              size ) _Generic(arr, char *:  charQuickSort, int *:  intQuickSort, float *↩
:floatQuickSort, double *:  doubleQuickSort, void **:  ptrQuickSort)(arr, size)
```

Quicksort 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	Pointer to the array to be sorted
SUCCESS	Array successfully sorted

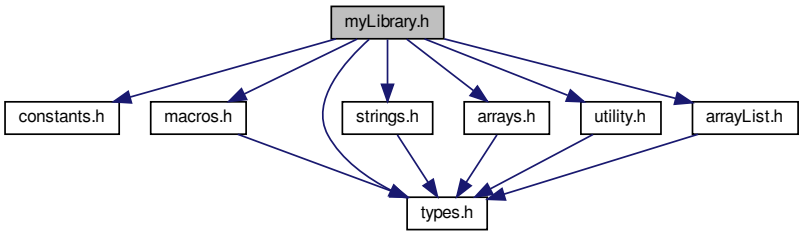
5.5 myLibrary.h File Reference

Includes all other headers. Useful for rapid import.

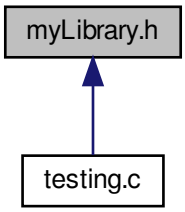
```
#include "constants.h"
#include "macros.h"
#include "types.h"
#include "strings.h"
#include "arrays.h"
#include "utility.h"
```

```
#include "arrayList.h"
```

Include dependency graph for myLibrary.h:



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



5.5.1 Detailed Description

Includes all other headers. Useful for rapid import.

Author

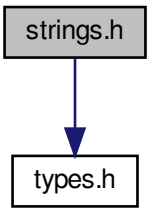
Pietro Firpo (pietro.firpo@pm.me)

5.6 strings.h File Reference

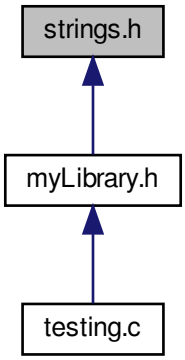
Common tasks with strings.

```
#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.

5.6.1 Detailed Description

Common tasks with strings.

Author

Pietro Firpo (pietro.firpo@pm.me)

5.6.2 Function Documentation

5.6.2.1 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

5.6.2.2 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

5.6.2.3 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	At least one among given pointers was null

5.6.2.4 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	At least one among given pointers was null

5.6.2.5 getString()

```
string getString ( )
```

Reads from terminal a string of arbitrary length.

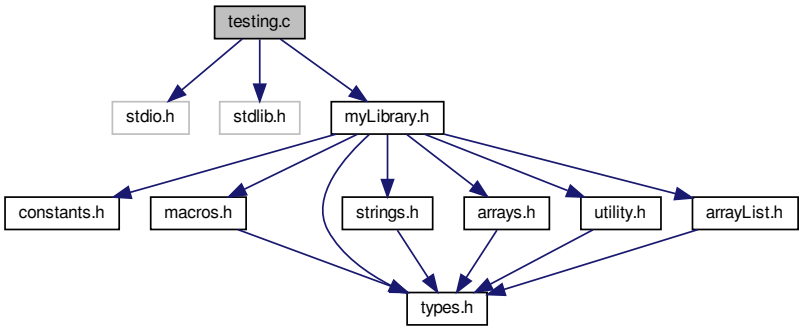
Returns

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

5.7 testing.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "myLibrary.h"
```

Include dependency graph for testing.c:



Functions

- int main ()

5.7.1 Function Documentation

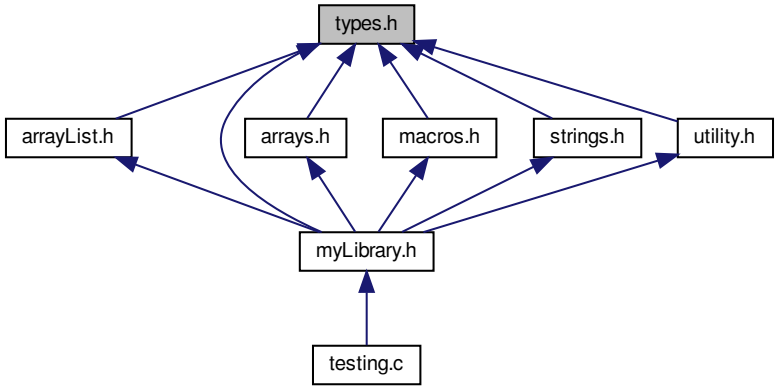
5.7.1.1 main()

```
int main ( )
```

5.8 types.h File Reference

Collection of useful types.

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



Data Structures

- struct ArrayList

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.*

5.8.1 Detailed Description

Collection of useful types.

Author

Pietro Firpo (pietro.firpo@pm.me)

5.8.2 Typedef Documentation

5.8.2.1 byte

```
typedef char byte
```

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

5.8.2.2 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

5.8.2.3 string

```
typedef char* string
```

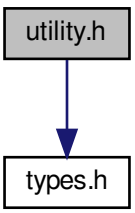
Alias for char *, used when an array of char is actually used as a string.

5.9 utility.h File Reference

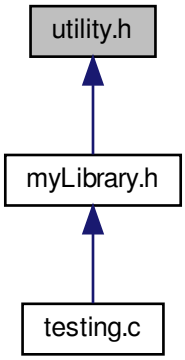
Common tasks such as comparing variables, swap bools, allocate memory.

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

Include dependency graph for utility.h:



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



Functions

- void checkCondition (byte condition, string errorString)
- 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.

5.9.1 Detailed Description

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

Author

Pietro Firpo (pietro.firpo@pm.me)

5.9.2 Function Documentation

5.9.2.1 byteCmp()

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

Compare two bytes.

Equivalent to `charCmp(a, b)`. Refer to charCmp().

5.9.2.2 charCmp()

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

Compare two chars.

Equivalent to `chooseCmp("%c", a, b)`. Refer to chooseCmp()

5.9.2.3 checkCondition()

```
void checkCondition (
            byte condition,
            string errorString )
```

5.9.2.4 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	At least one among given pointers was null

5.9.2.5 doubleCmp()

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

Compare two doubles.

Equivalent to `chooseCmp("%lf", a, b)`. Refer to chooseCmp()

5.9.2.6 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	At least one among given pointers was null

5.9.2.7 floatCmp()

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

Compare two floats.

Equivalent to `chooseCmp("%f", a, b)`. Refer to chooseCmp()

5.9.2.8 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

5.9.2.9 intCmp()

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

Compare two ints.

Equivalent to `chooseCmp("%i", a, b)`. Refer to chooseCmp()

5.9.2.10 ptrCmp()

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

Compare two pointers.

Equivalent to `chooseCmp("%p", a, b)`. Refer to chooseCmp()

5.9.2.11 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

5.9.2.12 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

5.9.2.13 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	At least one among given pointers was null

Index