Fondamenti di Informatica II e Lab.

La libreria <stdlib.h>

Pagine estratte dal sito www.cplusplus.com

Ultimo aggiornamento 02/11/2016

macro

NULL

Null pointer

This macro expands to a *null pointer constant*.

A *null-pointer constant* is an integral constant expression that evaluates to zero (like `0` or `0L`), or the cast of such value to type `void*` (like `(void*)0`).

A null pointer constant can be converted to any *pointer type* (or *pointer-to-member type*), which acquires a *null pointer value*. This is a special value that indicates that the pointer is not pointing to any object.

type

size_t

Unsigned integral type

Alias of one of the fundamental unsigned integer types.

It is a type able to represent the size of any object in bytes: size_t is the type returned by the sizeof operator and is widely used in the standard library to represent sizes and counts.

In <cstdlib>, it is used as the type of some parameters in the functions bsearch, qsort, calloc, malloc, realloc, mblen,mbtowc, mbstowcs and wcstombs, and in the case of mbstowcs and wcstombs also as its returning type.
In all cases it is used as a type to represent the length or count in bytes of a specific buffer or string.

function

malloc

```
void* malloc (size_t size);
```

Allocate memory block

Allocates a block of *size* bytes of memory, returning a pointer to the beginning of the block.

The content of the newly allocated block of memory is not initialized, remaining with indeterminate values.

If *size* is zero, the return value depends on the particular library implementation (it may or may not be a *null pointer*), but the returned pointer shall not be dereferenced.

Parameters

size

Size of the memory block, in bytes.
size_t is an unsigned integral type.

Return Value

On success, a pointer to the memory block allocated by the function.
The type of this pointer is always `void*`, which can be cast to the desired type of data pointer in order to be dereferenceable.
If the function failed to allocate the requested block of memory, a *null pointer* is returned.

function

free

```
void free (void* ptr);
```

Deallocate memory block

A block of memory previously allocated by a call to malloc, calloc or realloc is deallocated, making it available again for further allocations.

If *ptr* does not point to a block of memory allocated with the above functions, it causes *undefined behavior*.

If *ptr* is a *null pointer*, the function does nothing.

Notice that this function does not change the value of *ptr* itself, hence it still points to the same (now invalid) location.

Parameters

ptr

Pointer to a memory block previously allocated with malloc, calloc or realloc.

Return Value

none

function

calloc

```
void* calloc (size_t num, size_t size);
```

Allocate and zero-initialize array
Allocates a block of memory for an array of *num* elements, each of them *size* bytes long, and initializes all its bits to zero.

The effective result is the allocation of a zero-initialized memory block of `(num*size)` bytes.

If *size* is zero, the return value depends on the particular library implementation (it may or may not be a *null pointer*), but the returned pointer shall not be dereferenced.

Parameters

num
: Number of elements to allocate.

size
: Size of each element.

size_t is an unsigned integral type.

Return Value

On success, a pointer to the memory block allocated by the function.
The type of this pointer is always `void*`, which can be cast to the desired type of data pointer in order to be dereferenceable.
If the function failed to allocate the requested block of memory, a *null pointer* is returned.

function

realloc

```
void* realloc (void* ptr, size_t size);
```

Reallocate memory block

Changes the size of the memory block pointed to by *ptr*.

The function may move the memory block to a new location (whose address is returned by the function).

The content of the memory block is preserved up to the lesser of the new and old sizes, even if the block is moved to a new location. If the new *size* is larger, the value of the newly allocated portion is indeterminate.

In case that *ptr* is a null pointer, the function behaves like malloc, assigning a new block of *size* bytes and returning a pointer to its beginning.

Otherwise, if *size* is zero, the memory previously allocated at *ptr* is deallocated as if a call to free was made, and a *null pointer* is returned.

If the function fails to allocate the requested block of memory, a null pointer is returned, and the memory block pointed to by argument *ptr* is not deallocated (it is still valid, and with its contents unchanged).

Parameters

ptr

Pointer to a memory block previously allocated with malloc, calloc or realloc.
Alternatively, this can be a *null pointer*, in which case a new block is allocated (as if malloc was called).

size

New size for the memory block, in bytes.
size_t is an unsigned integral type.

Return Value

A pointer to the reallocated memory block, which may be either the same as *ptr* or a new location.
The type of this pointer is `void*`, which can be cast to the desired type of data pointer in order to be dereferenceable.
A *null-pointer* indicates either that *size* was zero (and thus *ptr* was deallocated), or that the function did not allocate storage (and thus the block pointed by *ptr* was not modified).

function

abs

```
int abs (int n);
```

Absolute value

Returns the absolute value of parameter *n* (/n/).

In C++, this function is also overloaded in header `<cmath>` for floating-point types (see cmath abs), in header `<complex>` for complex numbers (see complex abs), and in header `<valarray>` for valarrays (see valarray abs).

Parameters

n

Integral value.

Return Value

The absolute value of n.