Search: Go

Reference <cstdlib> malloc

Not logged in

register log in

C++

Information
Tutorials
Reference
Articles
Forum

Reference

C library:
<cassert> (assert.h)
<cctype> (ctype.h)
<cerrno> (errno.h)
<cfenv> (fenv.h)
<cfloat> (float.h)
<cinttypes> (inttypes.h)
<ciso646> (iso646.h)
<climits> (limits.h)
<clocale> (locale.h)
<cmath> (math.h)
<csetjmp> (setjmp.h)
<csignal> (signal.h)
<cstdarg> (stdarg.h)
<cstdbool> (stdbool.h)
<cstddef> (stddef.h)
<cstdint> (stdint.h)
<cstdio> (stdio.h)
<cstdlib> (stdlib.h)
<cstring> (string.h)
<ctgmath> (tgmath.h)
<ctime> (time.h)
<cuchar> (uchar.h)
<cwchar> (wchar.h)
<cwctype> (wctype.h)
Containers:
Input/Output:
Multi-threading:
Other:

<cstdlib> (stdlib.h)

functions:
abort
abs
atexit
atof
atoi
atol
atoll
at_quick_exit
bsearch
calloc
div
exit
free
getenv
labs
ldiv
llabs
lldiv
malloc
mblen
mbstowcs
mbtowc
qsort
quick_exit
rand
realloc
srand
strtod
strtof
strtol
strtold
strtoll
strtoul
strtoull
system
wcstombs
wctomb
_Exit
functions (non-standard):
itoa
types:
div_t
ldiv_t
lldiv_t
size_t

function

malloc

<cstdlib>

`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.

Example

```
/* malloc example: random string generator*/
#include <stdio.h>      /* printf, scanf, NULL */
#include <stdlib.h>     /* malloc, free, rand */

int main ()
{
  int i,n;
  char * buffer;

  printf ("How long do you want the string? ");
  scanf ("%d", &i);

  buffer = (char*) malloc (i+1);
  if (buffer==NULL) exit (1);

  for (n=0; n<i; n++)
    buffer[n]=rand()%26+'a';
  buffer[i]='\0';

  printf ("Random string: %s\n",buffer);
  free (buffer);

  return 0;
}
```

This program generates a string of the length specified by the user and fills it with alphabetic characters. The possible length of this string is only limited by the amount of memory available to `malloc`

Data races

Only the storage referenced by the returned pointer is modified. No other storage locations are accessed by the call.
If the function reuses the same unit of storage released by a *deallocation function* (such as `free` or `realloc`), the functions are synchronized in such a way that the deallocation happens entirely before the next allocation.

Exceptions (C++)

No-throw guarantee: this function never throws exceptions.

See also

free	Deallocate memory block (function)
calloc	Allocate and zero-initialize array (function)
realloc	Reallocate memory block (function)

macro constants:

- EXIT_FAILURE
- EXIT_SUCCESS
- MB_CUR_MAX
- NULL
- RAND_MAX

Home page | Privacy policy

Spotted an error? contact us