malloc

Defined in header `<stdlib.h>`

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

Allocates size bytes of uninitialized storage.

If allocation succeeds, returns a pointer that is suitably aligned for any object type with fundamental alignment.

If size is zero, the behavior of malloc is implementation-defined. For example, a null pointer may be returned. Alternatively, a non-null pointer may be returned; but such a pointer should not be dereferenced, and should be passed to free to avoid memory leaks.

malloc is thread-safe: it behaves as though only accessing the memory locations visible through its argument, and not any static storage.

A previous call to free or realloc that deallocates a region of memory *synchronizes-with* a call to malloc that allocates the same or a part of the same region of memory. This synchronization occurs after any access to the memory by the deallocating function and before any access to the memory by malloc. There is a single total order of all allocation and deallocation functions operating on each particular region of memory. (since C11)

Parameters

size - number of bytes to allocate

Return value

On success, returns the pointer to the beginning of newly allocated memory. To avoid a memory leak, the returned pointer must be deallocated with free() or realloc().

On failure, returns a null pointer.

Example

Run this code

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

int main(void)
{
    int *p1 = malloc(4*sizeof(int));  // allocates enough for an array of 4 int
    int *p2 = malloc(sizeof(int[4])); // same, naming the type directly
    int *p3 = malloc(4*sizeof *p3);   // same, without repeating the type name

    if(p1) {
        for(int n=0; n<4; ++n) // populate the array
            p1[n] = n*n;
        for(int n=0; n<4; ++n) // print it back out
            printf("p1[%d] == %d\n", n, p1[n]);
    }

    free(p1);
    free(p2);
    free(p3);
}
```

Output:

```
p1[0] == 0
p1[1] == 1
p1[2] == 4
p1[3] == 9
```

References

- C17 standard (ISO/IEC 9899:2018):
  - 7.22.3.4 The malloc function (p: 254)
- C11 standard (ISO/IEC 9899:2011):
  - 7.22.3.4 The malloc function (p: 349)
- C99 standard (ISO/IEC 9899:1999):
  - 7.20.3.3 The malloc function (p: 314)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 4.10.3.3 The malloc function

See also

C++ documentation for **malloc**