# free

Defined in header <stdlib.h>

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

Deallocates the space previously allocated by malloc(), calloc(), aligned_alloc(), (since C11) or realloc().

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

The behavior is undefined if the value of ptr does not equal a value returned earlier by malloc(), calloc(), realloc(), or aligned_alloc() (since C11).

The behavior is undefined if the memory area referred to by ptr has already been deallocated, that is, free() or realloc() has already been called with ptr as the argument and no calls to malloc(), calloc() or realloc() resulted in a pointer equal to ptr afterwards.

The behavior is undefined if after free() returns, an access is made through the pointer ptr (unless another allocation function happened to result in a pointer value equal to ptr)

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

A call to **free** that deallocates a region of memory *synchronizes-with* a call to any subsequent allocation function 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 the allocation function. There is a single total order of all allocation and deallocation functions operating on each particular region of memory. (since C11)

## Parameters

**ptr** - pointer to the memory to deallocate

## Return value

(none)

## Notes

The function accepts (and does nothing with) the null pointer to reduce the amount of special-casing. Whether allocation succeeds or not, the pointer returned by an allocation function can be passed to free()

## Example

Run this code

```
#include <stdlib.h>

int main(void)
{
    int *p1 = malloc(10*sizeof *p1);
    free(p1); // every allocated pointer must be freed

    int *p2 = calloc(10, sizeof *p2);
    int *p3 = realloc(p2, 1000*sizeof *p3);
    if(p3) // p3 not null means p2 was freed by realloc
       free(p3);
    else // p3 null means p2 was not freed
       free(p2);
}
```

## References

- C17 standard (ISO/IEC 9899:2018):
  - 7.22.3.3 The free function (p: 254)
- C11 standard (ISO/IEC 9899:2011):

- 7.22.3.3 The free function (p: 348)

- C99 standard (ISO/IEC 9899:1999):
  - 7.20.3.2 The free function (p: 313)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 4.10.3.2 The free function

## See also

| **C++ documentation** for **free** |
| --- |