std::memset

Defined in header <cstring>

```
void* memset( void* dest, int ch, std::size_t count );
```

Converts the value ch to unsigned char and copies it into each of the first count characters of the object pointed to by dest. If the object is a potentially-overlapping subobject or is not *TriviallyCopyable* (e.g., scalar, C-compatible struct, or an array of trivially copyable type), the behavior is undefined. If count is greater than the size of the object pointed to by dest, the behavior is undefined.

Parameters

dest	-	pointer to the object to fill
ch	-	fill byte
count	-	number of bytes to fill

Return value

dest

Notes

std::memset may be optimized away (under the as-if rules) if the object modified by this function is not accessed again for the rest of its lifetime (e.g. gcc bug 8537 (https://gcc.gnu.org/bugzilla/show_bug.cgi?id=8537)). For that reason, this function cannot be used to scrub memory (e.g. to fill an array that stored a password with zeroes). Solutions for that include std::fill with volatile pointers, C11 memset_s, FreeBSD explicit_bzero (https://www.freebsd.org/cgi/man.cgi?query=explicit_bzero) or Microsoft SecureZeroMemory (https://msdn.microsoft.com/en-us/library/windows/desktop/aa366877.aspx) .

Example

Run this code

```cpp
#include <iostream>
#include <cstring>

int main()
{
    int a[20];
    std::memset(a, 0, sizeof a);
    for (int ai : a) std::cout << ai;
}
```

Output:

```
00000000000000000000
```

See also

memcpy	copies one buffer to another (function)
memmove	moves one buffer to another (function)
wmemset	copies the given wide character to every position in a wide character array (function)
fill	copy-assigns the given value to every element in a range (function template)
fill_n	copy-assigns the given value to N elements in a range (function template)
is_trivially_copyable (C++11)	checks if a type is trivially copyable (class template)
C documentation for **memset**	