std::size_t

```
Defined in header <cstddef>
Defined in header <cstdio>
Defined in header <cstdlib>
Defined in header <cstring>
Defined in header <ctime>
Defined in header <cuchar>                          (since C++17)
Defined in header <cwchar>
```

```
typedef /*implementation-defined*/ size_t;
```

std::size_t is the unsigned integer type of the result of the sizeof operator as well as the sizeof... operator and the alignof operator (since C++11).

The bit width of std::size_t is not less than 16. (since C++11)

Notes

std::size_t can store the maximum size of a theoretically possible object of any type (including array). A type whose size cannot be represented by std::size_t is ill-formed (since C++14) On many platforms (an exception is systems with segmented addressing) std::size_t can safely store the value of any non-member pointer, in which case it is synonymous with std::uintptr_t.

std::size_t is commonly used for array indexing and loop counting. Programs that use other types, such as `unsigned int`, for array indexing may fail on, e.g. 64-bit systems when the index exceeds UINT_MAX or if it relies on 32-bit modular arithmetic.

When indexing C++ containers, such as std::string, std::vector, etc, the appropriate type is the member typedef `size_type` provided by such containers. It is usually defined as a synonym for std::size_t.

The integer literal suffix for std::size_t is uz (or UZ). (since C++23)

Example

Run this code

```
#include <cstddef>
#include <iostream>
#include <array>

int main()
{
    std::array<std::size_t, 10> a;

    // Example with C++23 size_t literal
    for (auto i = 0uz; i != a.size(); ++i)
        std::cout << (a[i] = i) << ' ';
    std::cout << '\n';

    // Example of decrementing loop
    for (std::size_t i = a.size(); i--;)
        std::cout << a[i] << ' ';

    // Note the naive decrementing loop:
    //  for (std::size_t i = a.size() - 1; i >= 0; --i) ...
    // is an infinite loop, because unsigned numbers are always non-negative

}
```

Output:

```
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
```

References

- C++20 standard (ISO/IEC 14882:2020):

  - 6.8.3 Compound types [basic.compound] (p: 75-76)
  - 7.6.2.5 Sizeof [expr.sizeof] (p: 129-130)
  - 7.6.2.6 Alignof [expr.alignof] (p: 130)
  - 17.2.4 Sizes, alignments, and offsets [support.types.layout] (p: 507-508)
- C++17 standard (ISO/IEC 14882:2017):
  - 6.9.2 Compound types [basic.compound] (p: 81-82)
  - 8.3.3 Sizeof [expr.sizeof] (p: 121-122)
  - 8.3.6 Alignof [expr.alignof] (p: 129)
  - 21.2.4 Sizes, alignments, and offsets [support.types.layout] (p: 479)
- C++14 standard (ISO/IEC 14882:2014):
  - 3.9.2 Compound types [basic.compound] (p: 73-74)
  - 5.3.3 Sizeof [expr.sizeof] (p: 109-110)
  - 5.3.6 Alignof [expr.alignof] (p: 116)
  - 18.2 Types [support.types] (p: 443-444)
- C++11 standard (ISO/IEC 14882:2011):
  - 5.3.3 Sizeof [expr.sizeof] (p: 111)
  - 5.3.6 Alignof [expr.alignof] (p: 116)
  - 18.2 Types [support.types] (p: 454-455)
- C++03 standard (ISO/IEC 14882:2003):
  - 5.3.3 Sizeof [expr.sizeof] (p: 79)
- C++98 standard (ISO/IEC 14882:1998):
  - 5.3.3 Sizeof [expr.sizeof] (p: 77)

See also

ptrdiff_t	signed integer type returned when subtracting two pointers (typedef)
offsetof	byte offset from the beginning of a standard-layout type to specified member (function macro)
integer literals	binary (C++14), decimal, octal, or hexadecimal numbers of integer type
C documentation for **size_t**	

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=cpp/types/size_t&oldid=134922"