sizeof operator

Queries size of the object or type.

Used when actual size of the object must be known.

Syntax

sizeof(*type* **)**	(1)
sizeof *expression*	(2)

Both versions are constant expressions of type std::size_t.

Explanation

1) Yields the size in bytes of the object representation of *type*.

2) Yields the size in bytes of the object representation of the type of *expression*, if that expression is evaluated.

Notes

Depending on the computer architecture, a byte   may consist of 8 *or more* bits, the exact number being recorded in CHAR_BIT.

The following sizeof expressions always evaluate to 1:

- sizeof(char)
- sizeof(signed char)
- sizeof(unsigned char)
- sizeof(std::byte) (since C++17)
- sizeof(char8_t) (since C++20)

sizeof cannot be used with function types, incomplete types, or bit-field glvalues.

When applied to a reference type, the result is the size of the referenced type.

When applied to a class type, the result is the number of bytes occupied by a complete object of that class, including any additional padding required to place such object in an array. The number of bytes occupied by a potentially-overlapping subobject may be less than the size of that object.

The result of sizeof is always nonzero, even if applied to an empty class type.

When applied to an expression, sizeof does not evaluate the expression, and even if the expression designates a polymorphic object, the result is the size of the static type of the expression. Lvalue-to-rvalue, array-to-pointer, or function-to-pointer conversions are not performed. Temporary materialization, however, is (formally) performed for prvalue arguments: the program is ill-formed if the argument is not destructible. (since C++17)

Keywords

sizeof

Example

The example output corresponds to a system with 64-bit pointers and 32-bit int.

Run this code

```
#include <iostream>

struct Empty {};
struct Base { int a; };
struct Derived : Base { int b; };
struct Bit { unsigned bit: 1; };
```

```
int main()
{
    Empty e;
    Derived d;
    Base& b = d;
    [[maybe_unused]] Bit bit;
    int a[10];
    std::cout
      << "1) size of empty class:              " << sizeof e        << '\n'
      << "2) size of pointer:                  " << sizeof &e       << '\n'
//    << "3) size of function:                 " << sizeof(void())  << '\n' // error
//    << "4) size of incomplete type:          " << sizeof(int[])   << '\n' // error
//    << "5) size of bit field:                " << sizeof bit.bit  << '\n' // error
      << "6) size of Bit class:                " << sizeof(Bit)     << '\n'
      << "7) size of array of 10 int:          " << sizeof(int[10]) << '\n'
      << "8) size of array of 10 int (2):      " << sizeof a        << '\n'
      << "9) length of array of 10 int:        " << ((sizeof a) / (sizeof *a))   << '\n'
      << "A) length of array of 10 int (2):    " << ((sizeof a) / (sizeof a[0])) << '\n'
      << "B) size of the Derived:              " << sizeof d        << '\n'
      << "C) size of the Derived through Base: " << sizeof b        << '\n';
}
```

Possible output:

```
1) size of empty class:              1
2) size of pointer:                  8
6) size of Bit class:                4
7) size of array of 10 int:          40
8) size of array of 10 int (2):      40
9) length of array of 10 int:        10
A) length of array of 10 int (2):    10
B) size of the Derived:              8
C) size of the Derived through Base: 4
```

See also

alignof operator(C++11)	queries alignment requirements of a type
sizeof... operator(C++11)	queries the number of elements in a parameter pack
C documentation for **sizeof**	

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=cpp/language/sizeof&oldid=132889"