public member function

std::ios_base::precision

<ios> <iostream>

```
get (1)  streamsize precision() const;
set (2)  streamsize precision (streamsize prec);
```

Get/Set floating-point decimal precision

The first form *(1)* returns the value of the current floating-point precision field for the stream.
The second form *(2)* also sets it to a new value.

The *floating-point precision* determines the maximum number of digits to be written on insertion operations to express floating-point values. How this is interpreted depends on whether the `floatfield` format flag is set to a specific notation (either `fixed` or `scientific`) or it is unset (using the *default notation*, which is not necessarily equivalent to either `fixed` nor `scientific`).

For the default locale:

- Using the default floating-point notation, the precision field specifies the maximum number of meaningful digits to display in total counting both those before and those after the decimal point. Notice that it is not a minimum, and therefore it does not pad the displayed number with trailing zeros if the number can be displayed with less digits than the *precision*.
- In both the `fixed` and `scientific` notations, the precision field specifies exactly how many digits to display after the decimal point, even if this includes trailing decimal zeros. The digits before the decimal point are not relevant for the *precision* in this case.

This *decimal precision* can also be modified using the parameterized manipulator `setprecision`.

Parameters

prec
: New value for the floating-point precision.
`streamsize` is a signed integral value.

Return Value

The *precision* selected in the stream before the call.

Example

```
// modify precision
#include <iostream>     // std::cout, std::ios

int main () {
  double f = 3.14159;
  std::cout.unsetf ( std::ios::floatfield );                // floatfield not set
  std::cout.precision(5);
  std::cout << f << '\n';

  std::cout.precision(10);
  std::cout << f << '\n';
  std::cout.setf( std::ios::fixed, std:: ios::floatfield ); // floatfield set to fixed
  std::cout << f << '\n';
  return 0;
}
```

Possible output:

```
3.1416
3.14159
3.1415900000
```

Notice how the first number written is just 5 digits long, while the second is 6, but not more, even though the stream's precision is now 10. That is because `precision` with the default `floatfield` only specifies the *maximum* number of digits to be displayed, but not the minimum.
The third number printed displays 10 digits after the decimal point because the `floatfield` format flag is in this case set to `fixed`.

Data races

Accesses *(1)* or modifies *(2)* the stream object.
Concurrent access to the same stream object may cause data races.

ios_base::width
ios_base::xalloc
member types:
ios_base::event
ios_base::event_callback
ios_base::failure
ios_base::fmtflags
ios_base::Init
ios_base::iostate
ios_base::openmode
ios_base::seekdir

Exception safety

Basic guarantee: if an exception is thrown, the stream is in a valid state.

See also

setprecision	Set decimal precision (function)
ios_base::width	Get/set field width (public member function)
ios_base::setf	Set specific format flags (public member function)