C++
Information
Tutorials
Reference
Articles
Forum

Reference
C library:
Containers:
Input/Output:
<fstream>
<iomanip>
<ios>
<iosfwd>
<iostream>
<istream>
<ostream>
<sstream>
<streambuf>
Multi-threading:
Other:

<ios>
types:
basic_ios
fpos
ios
ios_base
io_errc
streamoff
streampos
streamsize
wios
wstreampos
manipulators:
boolalpha
dec
defaultfloat
fixed
hex
hexfloat
internal
left
noboolalpha
noshowbase
noshowpoint
noshowpos
noskipws
nounitbuf
nouppercase
oct
right
scientific
showbase
showpoint
showpos
skipws
unitbuf
uppercase
other functions:
iostream_category

ios_base
ios_base::ios_base
ios_base::~ios_base
member functions:
ios_base::flags
ios_base::getloc
ios_base::imbue
ios_base::iword
ios_base::precision
ios_base::pword
ios_base::register_callback
ios_base::setf
ios_base::sync_with_stdio
ios_base::unsetf

public member type

std::ios_base::fmtflags

<ios> <iostream>

Type for stream format flags

Bitmask type to represent stream *format flags*.
This type is used as its parameter and/or return value by the member functions `flags`, `setf` and `unsetf`.

The values passed and retrieved by these functions can be any valid combination of the following member constants:

field	member constant	effect when set
independent flags	`boolalpha`	read/write bool elements as alphabetic strings (`true` and `false`).
	`showbase`	write integral values preceded by their corresponding numeric base prefix.
	`showpoint`	write floating-point values including always the decimal point.
	`showpos`	write non-negative numerical values preceded by a plus sign (+).
	`skipws`	skip leading whitespaces on certain input operations.
	`unitbuf`	flush output after each inserting operation.
	`uppercase`	write uppercase letters replacing lowercase letters in certain insertion operations.
numerical base (`basefield`)	`dec`	read/write integral values using decimal base format.
	`hex`	read/write integral values using hexadecimal base format.
	`oct`	read/write integral values using octal base format.
float format (`floatfield`)	`fixed`	write floating point values in fixed-point notation.
	`scientific`	write floating-point values in scientific notation.
adjustment (`adjustfield`)	`internal`	the output is padded to the *field width* by inserting *fill characters* at a specified internal point.
	`left`	the output is padded to the *field width* appending *fill characters* at the end.
	`right`	the output is padded to the *field width* by inserting *fill characters* at the beginning.

Three additional bitmask constants made of the combination of the values of each of the three groups of selective flags can also be used:

flag value	equivalent to
`adjustfield`	`left | right | internal`
`basefield`	`dec | oct | hex`
`floatfield`	`scientific | fixed`

The values of these constants can be combined into a single `fmtflags` value using the OR bitwise operator (|).

These constants are defined as public members in the `ios_base` class. Therefore, they can be refered to either directly by their name as `ios_base` members (like `ios_base::hex`) or by using any of their inherited classes or instantiated objects, like for example `ios::left` or `cout.oct`.

These values of type `ios_base::fmtflags` should not be confused with the manipulators that have the same name but in the global scope, because they are used in different circumstances. The manipulators cannot be used as values for `ios_base::fmtflags`, as well as these constants shouldn't be used instead of the manipulators. Notice the difference:

```
ios_base::skipws     // constant value of type ios_base::fmtflags
skipws               // manipulator (global function)
```

Notice that several manipulators have the same name as these member constants (but as global functions instead) - see manipulators. The behavior of these manipulators generally corresponds to the same as setting or unsetting them with `ios_base::setf` or `ios_base::unsetf`, but they should not be confused! Manipulators are global functions and these constants are member constants. For example, `showbase` is a manipulator, while `ios_base::showbase` is a constant value that can be used as parameter with `ios_base::setf`.

Example

```
// using ios_base::fmtflags
#include <iostream>     // std::cout, std::ios_base, std::ios,
                        // std::hex, std::showbase
int main () {

  // using fmtflags as class member constants:
  std::cout.setf (std::ios_base::hex , std::ios_base::basefield);
  std::cout.setf (std::ios_base::showbase);
  std::cout << 100 << '\n';

  // using fmtflags as inherited class member constants:

```

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


```
13  std::cout.setf (std::ios::hex , std::ios::basefield);
14  std::cout.setf (std::ios::showbase);
15  std::cout << 100 << '\n';
16
17  // using fmtflags as object member constants:
18  std::cout.setf (std::cout.hex , std::cout.basefield);
19  std::cout.setf (std::cout.showbase);
20  std::cout << 100 << '\n';
21
22  // using fmtflags as a type:
23  std::ios_base::fmtflags ff;
24  ff = std::cout.flags();
25  ff &= ~std::cout.basefield;   // unset basefield bits
26  ff |= std::cout.hex;          // set hex
27  ff |= std::cout.showbase;     // set showbase
28  std::cout.flags(ff);
29  std::cout << 100 << '\n';
30
31  // not using fmtflags, but using manipulators:
32  std::cout << std::hex << std::showbase << 100 << '\n';
33
34  return 0;
}
```

The code shows some different ways of printing the same result, using both the `fmtflags` member constants and their homonymous manipulators.

Output:

```
0x64
0x64
0x64
0x64
0x64
```

See also

ios_base::flags	Get/set format flags (public member function)
ios_base::setf	Set specific format flags (public member function)
ios_base::unsetf	Clear specific format flags (public member function)
setiosflags	Set format flags (function)