cplusplus.com
Search: Go
Not logged in
Reference <istream> istream operator>>
register log in

C++
Information
Tutorials
Reference
Articles
Forum

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

<istream>
class templates:
basic_iostream
basic_istream
classes:
iostream
istream
wiostream
wistream
manipulators:
ws

istream
istream::istream
istream::~istream
member classes:
istream::sentry
member functions:
istream::gcount
istream::get
istream::getline
istream::ignore
istream::operator>>
istream::peek
istream::putback
istream::read
istream::readsome
istream::seekg
istream::sync
istream::tellg
istream::unget
non-member overloads:
operator>> (istream)
protected members:
istream::operator= C++11
istream::swap C++11

public member function

std::istream::operator>>

<istream> <iostream>

C++98 C++11

```
                        istream& operator>> (bool& val);
                        istream& operator>> (short& val);
                        istream& operator>> (unsigned short& val);
                        istream& operator>> (int& val);
                        istream& operator>> (unsigned int& val);
arithmetic types (1)    istream& operator>> (long& val);
                        istream& operator>> (unsigned long& val);
                        istream& operator>> (float& val);
                        istream& operator>> (double& val);
                        istream& operator>> (long double& val);
                        istream& operator>> (void*& val);
  stream buffers (2)    istream& operator>> (streambuf* sb );
                        istream& operator>> (istream& (*pf)(istream&));
    manipulators (3)    istream& operator>> (ios& (*pf)(ios&));
                        istream& operator>> (ios_base& (*pf)(ios_base&));
```

Extract formatted input

This operator (>>) applied to an input stream is known as *extraction operator*. It is overloaded as a member function for:

(1) arithmetic types

Extracts and parses characters sequentially from the stream to interpret them as the representation of a value of the proper type, which is stored as the value of *val*.
Internally, the function accesses the input sequence by first constructing a `sentry` object (with *noskipws* set to `false`). Then (if `good`), it calls `num_get::get` (using the stream's *selected locale*) to perform both the extraction and the parsing operations, adjusting the stream's *internal state flags* accordingly. Finally, it destroys the `sentry` object before returning.

(2) stream buffers

Extracts as many characters as possible from the stream and inserts them into the output sequence controlled by the *stream buffer* object pointed by *sb* (if any), until either the input sequence is exhausted or the function fails to insert into the object pointed by *sb*.

C++98 C++11

The function is considered to perform *formatted input*: Internally, the function accesses the input sequence by first constructing a `sentry` object (with *noskipws* set to **`false`**). Then (if `good`), it extracts characters from its associated *stream buffer* object as if calling its member functions `sbumpc` or `sgetc`, and finally destroys the `sentry` object before returning.

(3) manipulators

Calls `pf(*this)`, where *pf* may be a *manipulator*.
Manipulators are functions specifically designed to be called when used with this operator.
This operation has no effect on the input sequence and extracts no characters (unless the manipulator itself does, like `ws`).

See `operator>>` for additional overloads (as non-member functions) of this operator.

Except where stated otherwise, calling this function does not alter the value returned by member `gcount`.

Parameters

`val`

Object where the value that the extracted characters represent is stored.
Notice that the type of this argument (along with the stream's *format flags*) influences what constitutes a valid representation.

`sb`

Pointer to a `streambuf` object on whose controlled output sequence the characters are copied.

`pf`

A function that takes and returns a stream object. It generally is a *manipulator function*.
The standard manipulators which have an effect when used on standard `istream` objects are:

manipulator	Effect
`ws`	Extracts whitespaces.
`boolalpha`/`noboolalpha`	Activates/deactivates the extraction of alphanumerical representations of values of type bool.
`skipws`/`noskipws`	Activates/deactivates whether leading whitespaces are discarded before formatted input operations.
`dec`/`hex`/`oct`	Sets that base used to interpret integral numerical values.

The following extended manipulators can also be applied to `istream` objects (these take additional arguments and require the explicit inclusion of the `<iomanip>` header):

manipulator	Effect
`setbase`	Sets the numerical base used to interpret integral numerical values.
`setiosflags`/`resetiosflags`	Set/reset format flags.

Return Value

The `istream` object (`*this`).

The extracted value or sequence is not returned, but directly stored in the variable passed as argument.

Errors are signaled by modifying the *internal state flags*, except for *(3)*, that never sets any flags (but the particular manipulator applied may):

flag	error
`eofbit`	The input sequence has no more characters available (*end-of-file* reached).
`failbit`	Either no characters were extracted, or the characters extracted could not be interpreted as a valid value of the appropriate type. For *(2)*, it is set when no characters are inserted in the object pointed by *sb*, or when *sb* is a *null pointer*.
`badbit`	Error on stream (such as when this function catches an exception thrown by an internal operation). When set, the integrity of the stream may have been affected.

Multiple flags may be set by a single operation.

If the operation sets an *internal state flag* that was registered with member `exceptions`, the function throws an exception of member type `failure`.

Example

```cpp
// example on extraction
#include <iostream>     // std::cin, std::cout, std::hex

int main () {
  int n;

  std::cout << "Enter a number: ";
  std::cin >> n;
  std::cout << "You have entered: " << n << '\n';

  std::cout << "Enter a hexadecimal number: ";
  std::cin >> std::hex >> n;         // manipulator
  std::cout << "Its decimal equivalent is: " << n << '\n';

  return 0;
}
```

This example demonstrates the use of some of the overloaded `operator>>` functions shown above using the standard `istream` object `cin`.

Data races

Modifies *val* or the object pointed by *sb*.
Modifies the stream object.
Concurrent access to the same stream object may cause data races, except for the standard stream object `cin` when this is *synchronized with stdio* (in this case, no data races are initiated, although no guarantees are given on the order in which extracted characters are attributed to threads).

Exception safety

Basic guarantee: if an exception is thrown, the object is in a valid state.
It throws an exception of member type `failure` if the resulting *error state flag* is not `goodbit` and member `exceptions` was set to throw for that state.
Any exception thrown by an internal operation is caught and handled by the function, setting `badbit`. If `badbit` was set on the last call to `exceptions`, the function rethrows the caught exception.

See also

istream::get	Get characters (public member function)
istream::getline	Get line (public member function)
ostream::operator<<	Insert formatted output (public member function)
istream::sentry	Prepare stream for input (public member class)