C++
Information
Tutorials
Reference
Articles
Forum

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

<ostream>
class templates:
basic_ostream
classes:
ostream
wostream
manipulators:
endl
ends
flush

ostream
ostream::ostream
ostream::~ostream
member classes:
ostream::sentry
member functions:
ostream::flush
ostream::operator<<
ostream::put
ostream::seekp
ostream::tellp
ostream::write
non-member overloads:
operator<< (ostream)
protected members:
ostream::operator= C++11
ostream::swap C++11

public member function

std::ostream::write

<ostream> <iostream>

```
ostream& write (const char* s, streamsize n);
```

Write block of data

Inserts the first *n* characters of the array pointed by *s* into the stream.

This function simply copies a block of data, without checking its contents: The array may contain *null characters*, which are also copied without stopping the copying process.

C++98 | C++11

Internally, the function accesses the output sequence by first constructing a `sentry` object.
Then (if `good`), it inserts character into its associated *stream buffer* object as if calling its member function `sputc` until *n* characters have been written or until an insertion fails (in this case it sets the `badbit` flag).
Finally, it destroys the `sentry` object before returning.

Parameters

s
: Pointer to an array of at least *n* characters.

n
: Number of characters to insert.
Integer value of type `streamsize` representing the size in characters of the block of data to write.
`streamsize` is a signed integral type.

Return Value

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

Errors are signaled by modifying the *internal state flags*:

flag	error
`eofbit`	-
`failbit`	May be set if the construction of `sentry` failed.
`badbit`	Either an insertion on the stream failed, or some other error happened (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

```
1  // Copy a file
2  #include <fstream>      // std::ifstream, std::ofstream
3
4  int main () {
5    std::ifstream infile ("test.txt",std::ifstream::binary);
6    std::ofstream outfile ("new.txt",std::ofstream::binary);
7
8    // get size of file
9    infile.seekg (0,infile.end);
10   long size = infile.tellg();
11   infile.seekg (0);
12
13   // allocate memory for file content
14   char* buffer = new char[size];
15
16   // read content of infile
17   infile.read (buffer,size);
18
19   // write to outfile
20   outfile.write (buffer,size);
21
22   // release dynamically-allocated memory
23   delete[] buffer;
24
25   outfile.close();
26   infile.close();
27   return 0;
28 }
```

This example copies a file into memory and then writes its content to a new file.

Data races

Access up to *n* characters pointed by *s*.
Modifies the stream object.
Concurrent access to the same stream object may cause data races, except for the standard stream objects (`cout`, `cerr`, `clog`) when these are *synchronized with stdio* (in this case, no data races are initiated, although no guarantees are given on the order in which characters from multiple threads are inserted).

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

ostream::put	Put character (public member function)
ostream::operator<<	Insert formatted output (public member function)
istream::read	Read block of data (public member function)