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

public member function

std::istream::read

<istream> <iostream>

```
istream& read (char* s, streamsize n);
```

Read block of data

Extracts *n* characters from the stream and stores them in the array pointed to by *s*.

This function simply copies a block of data, without checking its contents nor appending a *null character* at the end.

If the input sequence runs out of characters to extract (i.e., the end-of-file is reached) before *n* characters have been successfully read, the array pointed to by *s* contains all the characters read until that point, and both the `eofbit` and `failbit` flags are set for the stream.

Internally, the function accesses the input sequence by first constructing a `sentry` object (with *noskipws* set to `true`). 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.

The number of characters successfully read and stored by this function can be accessed by calling member `gcount`.

Parameters

s
: Pointer to an array where the extracted characters are stored.

n
: Number of characters to extract.
`streamsize` is a signed integral type.

Return Value

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

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

flag	error
`eofbit`	The function stopped extracting characters because the input sequence has no more characters available (*end-of-file* reached).
`failbit`	Either the function could not extract *n* characters or the construction of `sentry` failed.
`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

```
1  // read a file into memory
2  #include <iostream>     // std::cout
3  #include <fstream>      // std::ifstream
4
5  int main () {
6
7    std::ifstream is ("test.txt", std::ifstream::binary);
8    if (is) {
9      // get length of file:
10     is.seekg (0, is.end);
11     int length = is.tellg();
12     is.seekg (0, is.beg);
13
14     char * buffer = new char [length];
15
16     std::cout << "Reading " << length << " characters... ";
17     // read data as a block:
18     is.read (buffer,length);
19
20     if (is)
21       std::cout << "all characters read successfully.";
22     else
23       std::cout << "error: only " << is.gcount() << " could be read";
24     is.close();
25
26     // ...buffer contains the entire file...
27
```

```
28
29      delete[] buffer;
30    }
31    return 0;
  }
```

Possible output:

```
Reading 640 characters... all characters read successfully.
```

Data races

Modifies the elements in the array pointed to by *s* and 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::readsome	Read data available in buffer (public member function)
istream::operator>>	Extract formatted input (public member function)

Home page | Privacy policy
© cplusplus.com, 2000-2015 - All rights reserved - *v3.1*
Spotted an error? contact us