# Boost.Checks

Pierre Talbot

## Table of Contents

Boost.Checks ..... 2
- Overview ..... 2
  - Document Conventions ..... 2
- ISBN checking ..... 3
  - Synopsis ..... 3
- Universal Product Code (UPC) checking ..... 4
- International Article Number (EAN) checking ..... 4
- Tutorial Examples ..... 5
  - ISBN example ..... 5
- Hints and Tips ..... 5
- Acknowledgements ..... 5
- FAQs ..... 5
- References ..... 5
- Rationale ..... 6
- History ..... 6
- TODO ..... 6
- Version Info ..... 6

Checks Reference ..... 7
- Header <boost/checks/adler.hpp> ..... 7
- Header <boost/checks/amex.hpp> ..... 7
- Header <boost/checks/checks_fwd.hpp> ..... 7
- Header <boost/checks/crc.hpp> ..... 9
- Header <boost/checks/ean.hpp> ..... 9
- Header <boost/checks/EANcheck.cpp> ..... 13
- Header <boost/checks/EANcheck.hpp> ..... 13
- Header <boost/checks/fletcher.hpp> ..... 13
- Header <boost/checks/iban.hpp> ..... 13
- Header <boost/checks/IBMCheck.hpp> ..... 13
- Header <boost/checks/isan.hpp> ..... 13
- Header <boost/checks/isbn.hpp> ..... 13
- Header <boost/checks/ISBN_PAB.hpp> ..... 17
- Header <boost/checks/isbn_Vasconcelos.hpp> ..... 17
- Header <boost/checks/ISSN_PAB.hpp> ..... 19
- Header <boost/checks/luhn.hpp> ..... 19
- Header <boost/checks/mastercard.hpp> ..... 19
- Header <boost/checks/upc.hpp> ..... 19
- Header <boost/checks/UPCcheck.cpp> ..... 21
- Header <boost/checks/UPCcheck.hpp> ..... 21
- Header <boost/checks/verhoeff.hpp> ..... 21
- Header <boost/checks/visa.hpp> ..... 21
- Header <boost/checks/VISACheck.hpp> ..... 21

Class Index ..... 22

Typedef Index ..... 22

Function Index ..... 22

Macro Index ........ 23
Index ........ 23

# Boost.Checks

## Overview

This library provides a collection of functions for validating and creating check digits.

Most are primarily for checking the accuracy of short strings of typed input (though it obviously also provides against a mis-scan by a device like a bar code or card reader). The well-known ISBN is a typical example. All single altered digits, most double altered digits, and transpositions of two digits are caught, and the input rejected as an invalid ISBN.

**Important**

This is not (yet) an official Boost library. It was a Google Summer of Code project (2011) whose mentor organization was Boost. It remains a library under construction, the code is quite functional, but interfaces, library structure, and names may still be changed without notice. The current version is available at

**https://svn.boost.org/svn/boost/sandbox/SOC/2011/checks/libs/checks/doc/pdf/checks.pdf PDF documentation**

**https://svn.boost.org/svn/boost/sandbox/SOC/2011/checks/libs/checks/doc/html/index.html HTML documentation**

**https://svn.boost.org/svn/boost/sandbox/SOC/2011/checks/boost/checksboost Boost Sandbox checks source code**

[note Comments and suggestions (even bugs!) to Pierre Talbot pierre.talbot.6114(at) herslibramont (dot).be

### Document Conventions

- **Tutorials** are listed in the *Table of Contents* and include many examples that should help you get started quickly.
- **Source code** of the many *Examples* will often be your quickest start.
- **Reference section** prepared using Doxygen will provide the function and class signatures, but there is also an *index* of these.
- The main *index* will also help, especially if you know a word describing what it does, without needing to know the exact name chosen for the function.

This documentation makes use of the following naming and formatting conventions.

- C++ Code is in `fixed width font` and is syntax-highlighted.
- Other code is in `teletype fixed-width font`.
- Replaceable text that you will need to supply is in *`italics`*.
- If a name refers to a free function, it is specified like this: `free_function()`; that is, it is in code font and its name is followed by `()` to indicate that it is a free function.
- If a name refers to a class template, it is specified like this: `class_template<>`; that is, it is in code font and its name is followed by `<>` to indicate that it is a class template.
- If a name refers to a function-like macro, it is specified like this: `MACRO()`; that is, it is uppercase in code font and its name is followed by `()` to indicate that it is a function-like macro. Object-like macros appear without the trailing `()`.
- Names that refer to *concepts* in the generic programming sense are specified in CamelCase.

- Many code snippets assume an implicit namespace, for example, `std::` or `boost::checks`.
- If you have a feature request, or if it appears that the implementation is in error, please check the TODO section first, as well as the rationale section.

If you do not find your idea/complaint, please reach the author either through the Boost development list, or email the author(s) direct .

### Admonishments

**Note**

In addition, notes such as this one specify non-essential information that provides additional background or rationale.

**Tip**

These blocks contain information that you may find helpful while coding.

**Important**

These contain information that is imperative to understanding a concept. Failure to follow suggestions in these blocks will probably result in undesired behavior. Read all of these you find.

**Warning**

Failure to heed this will lead to incorrect, and very likely undesired, results.

## ISBN checking

The functions defined at <boost/checks/isbn.hpp> are for validating and computing check digits of International Standard Book Number (ISBN) strings.

### Error detecting

You probably want a section on how good things are at detecting changes in the string. This has been well studied for the Verhoeff/Gumm system.

Some of your tests might be devoted to confirming that this is really true?

All alterations of a single digit will be detected.

All alterations of two digits will be detected. ??? Is this right???

All two digit transpositions will be detected.

## Synopsis

```
is_isbn10 function
```

```
// This function checks if an `isbn10` is a valid ISBN.
bool is_isbn10(const std::string& isbn);

// This function computes and returns the check digit for a given 9 digit ISBN in `isbn`.
char isbn_check_digit(const std::string& isbn);
```

Both functions assume that `isbn` is a 10-digit ISBN containing only ANSI digits '0' to '9', or ANSI letters 'X', 'Y', 'x', 'y'.

## Universal Product Code (UPC) checking

UPC is a sub-set of International Article Number (EAN) - see Universal Product Code (UPC).

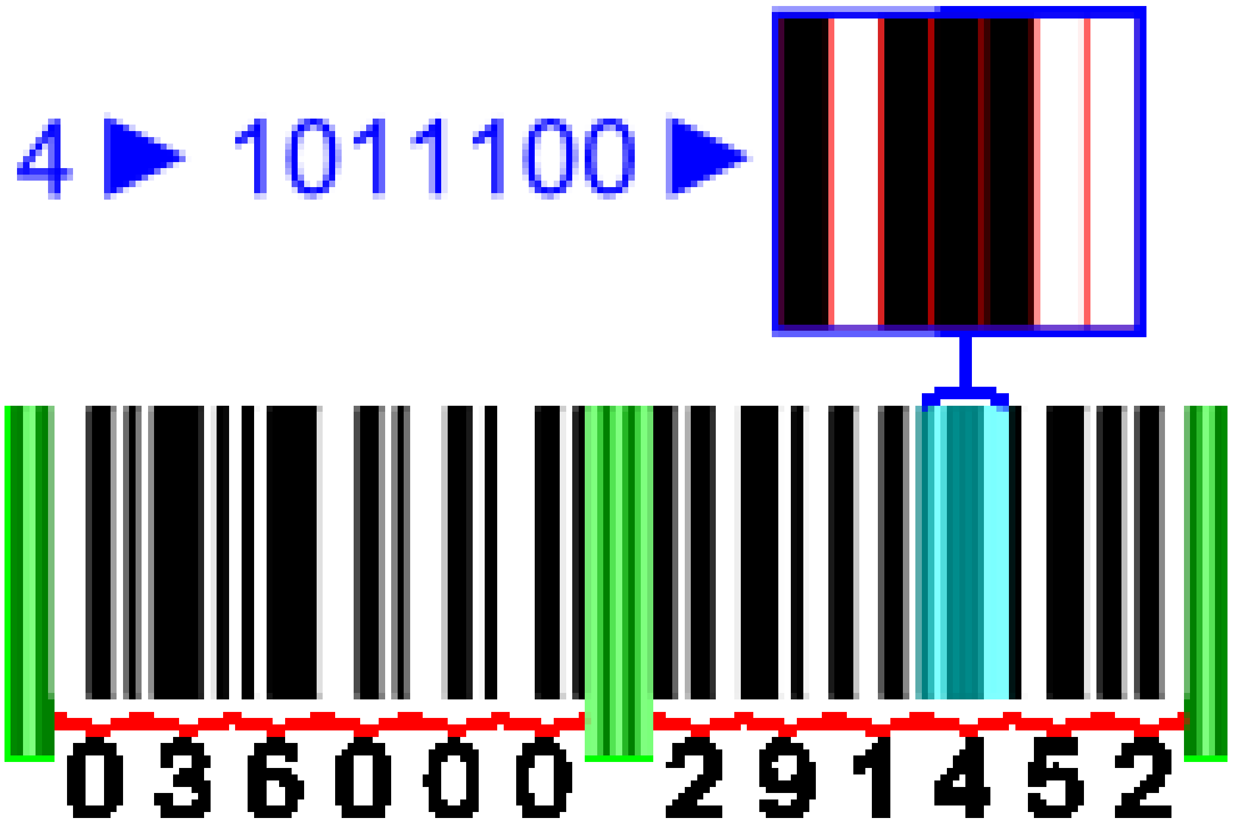
The original UPC is still in use and has 12 decimal digits, for example, a UPC for a box of tissues)

"03600029145X" where X is the check digit, in this case having a value of 2.

On products, it is usually printed as a barcode, but the decimal digits are visible too.

UPC EANUCC 12 barcode

This is a local copy of the barcode:



## International Article Number (EAN) checking

EAN is a super-set of the original US 12-digit Universal Product Code (UPC) UPC13 digit (12 + check digit) barcoding standard.

See ????

# Tutorial Examples

Here some general tutorial stuff.

followed by specific examples??

## ISBN example

Here is a really trivial example of making an ISBN check and also computing the check digit.

First we need to include the appropriate file including the check and compute functions.

```
#include <boost/checks/ISBN_PAB.hpp> // ISBN for books (Old PAB version using just string para↵
meter).
```

Then assuming that the ISBN is in a `std::string`, we can check a complete ISBN, and also compute the check digit from one lacking the check digit.

```
string s1 = "0201700735";

cout << "ISBN " << s1 << (ISBNcheck(s1) ? " is OK" : " is Wrong!") << endl;

string s2 = "020170073";

cout << "Check digit of " << s2 << " is " << ISBNcompute("020170073") << endl;
```

This provides this output:

```
ISBN 0201700735 is OK
Check digit of 020170073 is 5
```

# Hints and Tips

- This manual is also available as a single PDF file which may be easily emailed and printed.
- This is another tip?

# Acknowledgements

- Thanks to Google for providing the Summer of Code Program 2011 that enabled Pierre Talbot to write this library.
- UPC EANUCC 12 barcode is copied from Wikipedia under the Creative Commons license.

# FAQs

- Why are checks needed?
- How many alterations to the strings are detected (or undetected)?

# References

1. International Standard Book Number (ISBN)

2. International Standard Serial Number

## Rationale

This section records the rationale and compromises for some design decisions.

### Function parameter

- Functions that take a single `std::string` are convenient for users with simple requirements. This may be simplest what a string is typed in.
- Functions that use std:: iterators `begin` and `end` allow efficient use of a decimal digits string within other text, for example when a record is retrieve from a database.

### Scope of the project

- Scott McMurray has identifed four fairly distinct types of check:
  1. ISBN/ISSN/UPC/EAN/VISA/etc, for catching human-entry errors.
  2. hash functions as in hash tables, which only care about distribution.
  3. checksums like CRC32, for catching data transmission errors.
  4. and cryptographic hash functions, the only ones useful against malicious adversaries.

This project is directed first at the first class. Others might be the subject of future additions or other libraries.

- Performance is not a major objective, as most input is tiny, and the number of items often likely to be quite small.
- Convenience and flexibility for the user is the highest priority.

## History

1. Project started by Pierre Talbot June 2011 as a Google Summer of Code Project.
2. First release in Boost Sandbox for public comment ????

## TODO

This section lists items that are acknowledged as work still TODO.

1. Produce 1st version for comment.
2. Produce version for pre-review.

## Version Info

Last edit to Quickbook file I:\boost-sandbox\SOC\2011\checks\libs\checks\doc\checks.qbk was at 12:16:44 PM on 2011-Jun-17.

**Tip**

This version information should appear on the pdf version (but is redundant on html where the last revised date is on the bottom line of the home page).

**Warning**

Home page "Last revised" is GMT, not local time. Last edit date is local time.

**Caution**

It does not give the last edit date of other included .qbk files, so may mislead!

# Checks Reference

## Header <boost/checks/adler.hpp>

## Header <boost/checks/amex.hpp>

## Header <boost/checks/checks_fwd.hpp>

Boost.Checks forward declaration of function signatures.

This file can be used to copy a function signature, but is mainly provided for testing purposes.

```
template<typename In> bool Is_isbn10(In, In);
template<typename In> char isbn10_check_digit(In, In);
bool ISBNcheck(string s);
char ISBNcompute(string s);
```

## Function template Is_isbn10

Is_isbn10

## Synopsis

```
// In header: <boost/checks/checks_fwd.hpp>


template<typename In> bool Is_isbn10(In isbn_begin, In isbn_end);
```

### Description

Check the validity of the International Standard Book Number (ISBN) of size 10.

| | | |
|---|---|---|
| Parameters: | `isbn_begin` | Represents the beginning of the ISBN sequence to check. |
| | `isbn_end` | Represents one off the end of the ISBN sequence to check. |
| Requires: | isbn_begin and isbn_end are valid initialized iterators.The length of the sequence should be at least of size 10 and the sequence should contains only dash(es) and digits. | |
| Postconditions: | isbn_begin and isbn_end are inchanged. | |
| Returns: | true if the sequence is a valid ISBN of size 10, otherwise false. | |

## Function template isbn10_check_digit

isbn10_check_digit

## Synopsis

```
// In header: <boost/checks/checks_fwd.hpp>


template<typename In> char isbn10_check_digit(In isbn_begin, In isbn_end);
```

### Description

Compute the check digit of the International Standard Book Number (ISBN) of size 10.

| | | |
|---|---|---|
| Parameters: | `isbn_begin` | Represents the beginning of the ISBN sequence to check. |
| | `isbn_end` | Represents one off the end of the ISBN sequence to check. |
| Requires: | isbn_begin and isbn_end are valid initialized iterators. The length of the sequence should be of size 9 and the sequence should contains only digits and dashes. | |
| Postconditions: | isbn_begin and isbn_end are inchanged. | |
| Returns: | The check digit of the ISBN of size 9 provided, which can be between '0' and '9' or 'X'. Otherwise 0 is returned if the ISBN of size 9 provided is not correct. | |

## Header <boost/checks/crc.hpp>

## Header <boost/checks/ean.hpp>

```
namespace boost {
  namespace checks {
    template<typename In> char ean13_check_digit(In, In);
    template<typename In> char ean8_check_digit(In, In);
    template<typename In, typename Prefix>
      bool Is_ean13(In, In, Prefix = null, Prefix = null);
    template<typename In> bool Is_ean8(In, In);
  }
}
```

## Function template ean13_check_digit

boost::checks::ean13_check_digit

## Synopsis

```
// In header: <boost/checks/ean.hpp>


template<typename In> char ean13_check_digit(In ean_begin, In ean_end);
```

### Description

Compute the check digit of the European Article Numbering of size 13 (EAN-13) provided.

## Function template ean8_check_digit

boost::checks::ean8_check_digit

## Synopsis

```
// In header: <boost/checks/ean.hpp>


template<typename In> char ean8_check_digit(In ean_begin, In ean_end);
```

### Description

Compute the check digit of the European Article Numbering of size 8 (EAN-8) provided.

## Function template Is_ean13

boost::checks::Is_ean13

## Synopsis

```
// In header: <boost/checks/ean.hpp>


template<typename In, typename Prefix>
  bool Is_ean13(In ean_begin, In ean_end, Prefix ean_prefix_begin = null,
                Prefix ean_prefix_end = null);
```

### Description

Check the validity of the European Article Numbering of size 13 (EAN-13) provided.

| | | |
|---|---|---|
| Parameters: | `ean_begin` | Represents the beginning of the EAN sequence to check. |
| | `ean_end` | Represents one off the end of the EAN sequence to check. |
| | `ean_prefix_begin` | Represents the beginning of the prefixes sequence to allow. Default : null, all the prefixes are allowed. |
| | `ean_prefix_end` | Represents the ending of the prefixes sequence to allow. Default : null, all the prefixes are allowed. |
| Requires: | ean_begin and ean_end are valid initialized iterators. If ean_prefix_begin and ean_prefix_end are passed as arguments, they must be valid initialized iterators. | |
| Postconditions: | ean_begin, ean_end, ean_prefix_begin, and ean_prefix_end are unchanged. | |
| Returns: | True if the EAN delimited by ean_begin and ean_end is a valid EAN of size 13 with a prefix | |

## Function template Is_ean8

boost::checks::Is_ean8

## Synopsis

```
// In header: <boost/checks/ean.hpp>


template<typename In> bool Is_ean8(In ean_begin, In ean_end);
```

### Description

Check the validity of the European Article Numbering of size 8 (EAN-8) provided.

## Header <boost/checks/EANcheck.cpp>

```

bool EAN8check(std::string s);
char EAN8compute(std::string s);
bool EANcheck(std::string s);
char EANcompute(std::string s);
```

## Header <boost/checks/EANcheck.hpp>

```

bool EANcheck(std::string s);
char EANcompute(std::string s);
```

## Header <boost/checks/fletcher.hpp>

## Header <boost/checks/iban.hpp>

## Header <boost/checks/IBMCheck.hpp>

```

bool IBMcheck(string s);
char IBMcompute(string s);
```

## Header <boost/checks/isan.hpp>

## Header <boost/checks/isbn.hpp>

```
namespace boost {
  namespace checks {
    template<typename In> bool Is_isbn10(In, In);
    template<typename In> bool Is_isbn13(In, In);
    template<typename In> char isbn10_check_digit(In, In);
    template<typename In> char isbn13_check_digit(In, In);
  }
}
```

## Function template Is_isbn10

boost::checks::Is_isbn10

## Synopsis

```
// In header: <boost/checks/isbn.hpp>


template<typename In> bool Is_isbn10(In isbn_begin, In isbn_end);
```

### Description

Check the validity of the International Standard Book Number (ISBN) of size 10.

| | | |
|---|---|---|
| Parameters: | `isbn_begin` | Represents the beginning of the ISBN sequence to check. |
| | `isbn_end` | Represents one off the end of the ISBN sequence to check. |
| Requires: | isbn_begin and isbn_end are valid initialized iterators.The length of the sequence should be at least of size 10 and the sequence should contains only dash(es) and digits. | |
| Postconditions: | isbn_begin and isbn_end are inchanged. | |
| Returns: | true if the sequence is a valid ISBN of size 10, otherwise false. | |

## Function template Is_isbn13

boost::checks::Is_isbn13

## Synopsis

```
// In header: <boost/checks/isbn.hpp>


template<typename In> bool Is_isbn13(In isbn_begin, In isbn_end);
```

### Description

Check the validity of the International Standard Book Number (ISBN) of size 13. (It is a ISBN encapsulated into a EAN).

## Function template isbn10_check_digit

boost::checks::isbn10_check_digit

## Synopsis

```
// In header: <boost/checks/isbn.hpp>


template<typename In> char isbn10_check_digit(In isbn_begin, In isbn_end);
```

### Description

Compute the check digit of the International Standard Book Number (ISBN) of size 10.

| | | |
|---|---|---|
| Parameters: | `isbn_begin` | Represents the beginning of the ISBN sequence to check. |
| | `isbn_end` | Represents one off the end of the ISBN sequence to check. |
| Requires: | isbn_begin and isbn_end are valid initialized iterators. The length of the sequence should be of size 9 and the sequence should contains only digits and dashes. | |
| Postconditions: | isbn_begin and isbn_end are inchanged. | |
| Returns: | The check digit of the ISBN of size 9 provided, which can be between '0' and '9' or 'X'. Otherwise 0 is returned if the ISBN of size 9 provided is not correct. | |

## Function template isbn13_check_digit

boost::checks::isbn13_check_digit

## Synopsis

```
// In header: <boost/checks/isbn.hpp>


template<typename In> char isbn13_check_digit(In isbn_begin, In isbn_end);
```

### Description

Compute the check digit of the International Standard Book Number (ISBN) of size 13. (It is a ISBN encapulsed into a EAN).

## Header <boost/checks/ISBN_PAB.hpp>

Obselete versio of ISBN check and compute.

```

bool ISBNcheck(string s);
bool ISBNcheck(std::string s);
char ISBNcompute(std::string s);
char ISBNcompute(string s);
```

## Header <boost/checks/isbn_Vasconcelos.hpp>

```
namespace boost {
  bool is_isbn(const std::string &);
  char isbn_check_digit(const std::string &);
}
```

## Function is_isbn

boost::is_isbn

## Synopsis

```
// In header: <boost/checks/isbn_Vasconcelos.hpp>


bool is_isbn(const std::string & isbn);
```

### Description

This function checks if a `isbn' is a valid ISBN

## Function isbn_check_digit

boost::isbn_check_digit

## Synopsis

```
// In header: <boost/checks/isbn_Vasconcelos.hpp>


char isbn_check_digit(const std::string & isbn);
```

### Description

This function computes the check digit for a given ISBN in `isbn'

## Header <boost/checks/ISSN_PAB.hpp>

```

bool ISSNcheck(string s);
char ISSNcompute(string s);
```

## Header <boost/checks/luhn.hpp>

## Header <boost/checks/mastercard.hpp>

## Header <boost/checks/upc.hpp>

```
namespace boost {
  namespace checks {
    template<typename In> bool Is_upca(In, In);
    template<typename In> char upca_check_digit(In, In);
  }
}
```

## Function template Is_upca

boost::checks::Is_upca

## Synopsis

```
// In header: <boost/checks/upc.hpp>


template<typename In> bool Is_upca(In upc_begin, In upc_end);
```

### Description

Check the validity of the Universal Product Code category A (UPC-A) provided.

| | |
|---|---|
| Parameters: | `upc_begin` Represents the beginning of the UPC sequence to check. |
| Requires: | upc_begin and upc_end are valid initialized iterators. The length of the sequence should be of size 12 and the sequence should contains only digits. |
| Postconditions: | upc_begin and upc_end are unchanged. |
| Returns: | true if the UPC sequence is valid which it means that the check digit is equals to the last character. Otherwise false. |

## Function template upca_check_digit

boost::checks::upca_check_digit

## Synopsis

```
// In header: <boost/checks/upc.hpp>


template<typename In> char upca_check_digit(In upc_begin, In upc_end);
```

### Description

Compute the check digit of the Universal Product Code category A (UPC-A) provided /tparam Iterator which represents the beginning or the ending of a sequence of character. /param [in] upc_begin Represents the beginning of the UPC sequence to check. /param [in] upc_end Represents one off the end of the UPC sequence to check. /pre upc_begin and upc_end are valid initialized iterators. The length of the sequence should be of size 11 and the sequence should contains only digits. /post upc_begin and upc_end are unchanged. /result 0 if the check digit couldn't be calculated (Exemple : wrong size, ...). Otherwise, the check digit character between '0' and '9'.

## Header <boost/checks/UPCcheck.cpp>

```

bool UPCcheck(std::string s);
char UPCcompute(std::string s);
```

## Header <boost/checks/UPCcheck.hpp>

```

bool UPCcheck(std::string s);
char UPCcompute(std::string s);
```

## Header <boost/checks/verhoeff.hpp>

## Header <boost/checks/visa.hpp>

## Header <boost/checks/VISACheck.hpp>

```

bool VISAcheck(string s);
char VISAcompute(string s);
```

# Class Index

# Typedef Index

# Function Index

## E

ean13_check_digit
- Header < boost/checks/ean.hpp >, 9, 10

EAN8check
- Header < boost/checks/EANcheck.cpp >, 13

EAN8compute
- Header < boost/checks/EANcheck.cpp >, 13

ean8_check_digit
- Header < boost/checks/ean.hpp >, 9, 11

EANcheck
- Header < boost/checks/EANcheck.cpp >, 13
- Header < boost/checks/EANcheck.hpp >, 13

EANcompute
- Header < boost/checks/EANcheck.cpp >, 13
- Header < boost/checks/EANcheck.hpp >, 13

## I

IBMcheck
- Header < boost/checks/IBMCheck.hpp >, 13

IBMcompute
- Header < boost/checks/IBMCheck.hpp >, 13

isbn10_check_digit
- Header < boost/checks/checks_fwd.hpp >, 7, 9
- Header < boost/checks/isbn.hpp >, 13, 16

isbn13_check_digit
- Header < boost/checks/isbn.hpp >, 13, 17

ISBNcheck
- Header < boost/checks/checks_fwd.hpp >, 7
- Header < boost/checks/ISBN_PAB.hpp >, 17

ISBNcompute
- Header < boost/checks/checks_fwd.hpp >, 7
- Header < boost/checks/ISBN_PAB.hpp >, 17

isbn_check_digit
- Header < boost/checks/isbn_Vasconcelos.hpp >, 17, 19
- Synopsis, 3

ISSNcheck
- Header < boost/checks/ISSN_PAB.hpp >, 19

ISSNcompute
- Header < boost/checks/ISSN_PAB.hpp >, 19

Is_ean13
- Header < boost/checks/ean.hpp >, 9, 12

Is_ean8
- Header < boost/checks/ean.hpp >, 9, 13

is_isbn
- Header < boost/checks/isbn_Vasconcelos.hpp >, 17, 18

Is_isbn10
- Header < boost/checks/checks_fwd.hpp >, 7, 8

    Header < boost/checks/isbn.hpp >, 13, 14
Is_isbn13
    Header < boost/checks/isbn.hpp >, 13, 15
Is_upca
    Header < boost/checks/upc.hpp >, 19, 20

## U
upca_check_digit
    Header < boost/checks/upc.hpp >, 19, 21
UPCcheck
    Header < boost/checks/UPCcheck.cpp >, 21
    Header < boost/checks/UPCcheck.hpp >, 21
UPCcompute
    Header < boost/checks/UPCcheck.cpp >, 21
    Header < boost/checks/UPCcheck.hpp >, 21

## V
VISAcheck
    Header < boost/checks/VISACheck.hpp >, 21
VISAcompute
    Header < boost/checks/VISACheck.hpp >, 21

# Macro Index

# Index

## A
acknowledgements
    Acknowledgements, 5
Acknowledgements
    acknowledgements, 5
    C++, 5

## B
book
    Header < boost/checks/checks_fwd.hpp >, 8, 9
    Header < boost/checks/isbn.hpp >, 14, 15, 16, 17
    ISBN checking, 3
    References, 5
Boost.Checks
    C++, 1, 2
    index, 2
    version, 1

## C
C++
    Acknowledgements, 5
    Boost.Checks, 1, 2
    Checks Reference, 7
    Document Conventions, 2, 3
    FAQs, 5
    Header < boost/checks/adler.hpp >, 7
    Header < boost/checks/amex.hpp >, 7
    Header < boost/checks/checks_fwd.hpp >, 7, 8, 9

Header < boost/checks/crc.hpp >, 9
Header < boost/checks/ean.hpp >, 9, 10, 11, 12, 13
Header < boost/checks/EANcheck.cpp >, 13
Header < boost/checks/EANcheck.hpp >, 13
Header < boost/checks/fletcher.hpp >, 13
Header < boost/checks/iban.hpp >, 13
Header < boost/checks/IBMCheck.hpp >, 13
Header < boost/checks/isan.hpp >, 13
Header < boost/checks/isbn.hpp >, 13, 14, 15, 16, 17
Header < boost/checks/ISBN_PAB.hpp >, 17
Header < boost/checks/isbn_Vasconcelos.hpp >, 17, 18, 19
Header < boost/checks/ISSN_PAB.hpp >, 19
Header < boost/checks/luhn.hpp >, 19
Header < boost/checks/mastercard.hpp >, 19
Header < boost/checks/upc.hpp >, 19, 20, 21
Header < boost/checks/UPCcheck.cpp >, 21
Header < boost/checks/UPCcheck.hpp >, 21
Header < boost/checks/verhoeff.hpp >, 21
Header < boost/checks/visa.hpp >, 21
Header < boost/checks/VISACheck.hpp >, 21
History, 6
International Article Number (EAN) checking, 4
ISBN checking, 3
ISBN example, 5
Overview, 2
Rationale, 6
Synopsis, 3
TODO, 6
Tutorial Examples, 5
Universal Product Code (UPC) checking, 4
Version Info, 6
card
Overview, 2
Checks Reference
C++, 7

D
Document Conventions
C++, 2, 3
Doxygen, 2
example, 2, 3
index, 2
italic, 2
pre-conditions, 2
snippet, 3
Doxygen
Document Conventions, 2

E
ean13_check_digit
Header < boost/checks/ean.hpp >, 9, 10
EAN8check
Header < boost/checks/EANcheck.cpp >, 13
EAN8compute
Header < boost/checks/EANcheck.cpp >, 13
ean8_check_digit
Header < boost/checks/ean.hpp >, 9, 11

EANcheck
    Header < boost/checks/EANcheck.cpp >, 13
    Header < boost/checks/EANcheck.hpp >, 13
EANcompute
    Header < boost/checks/EANcheck.cpp >, 13
    Header < boost/checks/EANcheck.hpp >, 13
example
    Document Conventions, 2, 3
    ISBN example, 5
    Overview, 2
    Rationale, 6
    Tutorial Examples, 5
    Universal Product Code (UPC) checking, 4

## F

FAQs
    C++, 5

## G

Gumm
    ISBN checking, 3

## H

Header < boost/checks/adler.hpp >
    C++, 7
Header < boost/checks/amex.hpp >
    C++, 7
Header < boost/checks/checks_fwd.hpp >
    book, 8, 9
    C++, 7, 8, 9
    ISBN, 8, 9
    isbn10_check_digit, 7, 9
    ISBNcheck, 7
    ISBNcompute, 7
    Is_isbn10, 7, 8
    post-conditions, 8, 9
Header < boost/checks/crc.hpp >
    C++, 9
Header < boost/checks/ean.hpp >
    C++, 9, 10, 11, 12, 13
    ean13_check_digit, 9, 10
    ean8_check_digit, 9, 11
    Is_ean13, 9, 12
    Is_ean8, 9, 13
    post-conditions, 12
    pre-conditions, 9, 12
Header < boost/checks/EANcheck.cpp >
    C++, 13
    EAN8check, 13
    EAN8compute, 13
    EANcheck, 13
    EANcompute, 13
Header < boost/checks/EANcheck.hpp >
    C++, 13
    EANcheck, 13
    EANcompute, 13
Header < boost/checks/fletcher.hpp >

C++, 13
Header < boost/checks/iban.hpp >
C++, 13
Header < boost/checks/IBMCheck.hpp >
C++, 13
IBMcheck, 13
IBMcompute, 13
Header < boost/checks/isan.hpp >
C++, 13
Header < boost/checks/isbn.hpp >
book, 14, 15, 16, 17
C++, 13, 14, 15, 16, 17
ISBN, 13, 14, 15, 16, 17
isbn10_check_digit, 13, 16
isbn13_check_digit, 13, 17
Is_isbn10, 13, 14
Is_isbn13, 13, 15
post-conditions, 14, 16
Header < boost/checks/ISBN_PAB.hpp >
C++, 17
ISBN, 17
ISBNcheck, 17
ISBNcompute, 17
Header < boost/checks/isbn_Vasconcelos.hpp >
C++, 17, 18, 19
ISBN, 18, 19
isbn_check_digit, 17, 19
is_isbn, 17, 18
Header < boost/checks/ISSN_PAB.hpp >
C++, 19
ISSNcheck, 19
ISSNcompute, 19
Header < boost/checks/luhn.hpp >
C++, 19
Header < boost/checks/mastercard.hpp >
C++, 19
Mastercard, 19
Header < boost/checks/upc.hpp >
C++, 19, 20, 21
Is_upca, 19, 20
post-conditions, 20, 21
pre-conditions, 21
upca_check_digit, 19, 21
Header < boost/checks/UPCcheck.cpp >
C++, 21
UPCcheck, 21
UPCcompute, 21
Header < boost/checks/UPCcheck.hpp >
C++, 21
UPCcheck, 21
UPCcompute, 21
Header < boost/checks/verhoeff.hpp >
C++, 21
Verhoeff, 21
Header < boost/checks/visa.hpp >
C++, 21
VISA, 21
Header < boost/checks/VISACheck.hpp >

C++, 21
VISAcheck, 21
VISAcompute, 21
History
C++, 6

## I

IBMcheck
Header < boost/checks/IBMCheck.hpp >, 13
IBMcompute
Header < boost/checks/IBMCheck.hpp >, 13
index
Boost.Checks, 2
Document Conventions, 2
Overview, 2
International Article Number (EAN) checking
C++, 4
ISBN
Header < boost/checks/checks_fwd.hpp >, 8, 9
Header < boost/checks/isbn.hpp >, 13, 14, 15, 16, 17
Header < boost/checks/ISBN_PAB.hpp >, 17
Header < boost/checks/isbn_Vasconcelos.hpp >, 18, 19
ISBN checking, 3
ISBN example, 5
Overview, 2
Rationale, 6
References, 5
Synopsis, 3
ISBN checking
book, 3
C++, 3
Gumm, 3
ISBN, 3
Verhoeff, 3
ISBN example
C++, 5
example, 5
ISBN, 5
version, 5
isbn10_check_digit
Header < boost/checks/checks_fwd.hpp >, 7, 9
Header < boost/checks/isbn.hpp >, 13, 16
isbn13_check_digit
Header < boost/checks/isbn.hpp >, 13, 17
ISBNcheck
Header < boost/checks/checks_fwd.hpp >, 7
Header < boost/checks/ISBN_PAB.hpp >, 17
ISBNcompute
Header < boost/checks/checks_fwd.hpp >, 7
Header < boost/checks/ISBN_PAB.hpp >, 17
isbn_check_digit
Header < boost/checks/isbn_Vasconcelos.hpp >, 17, 19
Synopsis, 3
ISSN
Rationale, 6
ISSNcheck
Header < boost/checks/ISSN_PAB.hpp >, 19

ISSNcompute
    Header < boost/checks/ISSN_PAB.hpp >, 19
Is_ean13
    Header < boost/checks/ean.hpp >, 9, 12
Is_ean8
    Header < boost/checks/ean.hpp >, 9, 13
is_isbn
    Header < boost/checks/isbn_Vasconcelos.hpp >, 17, 18
Is_isbn10
    Header < boost/checks/checks_fwd.hpp >, 7, 8
    Header < boost/checks/isbn.hpp >, 13, 14
Is_isbn13
    Header < boost/checks/isbn.hpp >, 13, 15
Is_upca
    Header < boost/checks/upc.hpp >, 19, 20
italic
    Document Conventions, 2

**M**

Mastercard
    Header < boost/checks/mastercard.hpp >, 19

**O**

Overview
    C++, 2
    card, 2
    example, 2
    index, 2
    ISBN, 2
    version, 2

**P**

post-conditions
    Header < boost/checks/checks_fwd.hpp >, 8, 9
    Header < boost/checks/ean.hpp >, 12
    Header < boost/checks/isbn.hpp >, 14, 16
    Header < boost/checks/upc.hpp >, 20, 21
pre-conditions
    Document Conventions, 2
    Header < boost/checks/ean.hpp >, 9, 12
    Header < boost/checks/upc.hpp >, 21
    TODO, 6

**Q**

Quickbook
    Version Info, 6

**R**

Rationale
    C++, 6
    example, 6
    ISBN, 6
    ISSN, 6
    VISA, 6
References
    book, 5

ISBN, 5

S

snippet
Document Conventions, 3
Synopsis
C++, 3
ISBN, 3
isbn_check_digit, 3

T

TODO
C++, 6
pre-conditions, 6
version, 6
Tutorial Examples
C++, 5
example, 5

U

Universal Product Code (UPC) checking
C++, 4
example, 4
upca_check_digit
Header < boost/checks/upc.hpp >, 19, 21
UPCcheck
Header < boost/checks/UPCcheck.cpp >, 21
Header < boost/checks/UPCcheck.hpp >, 21
UPCcompute
Header < boost/checks/UPCcheck.cpp >, 21
Header < boost/checks/UPCcheck.hpp >, 21

V

Verhoeff
Header < boost/checks/verhoeff.hpp >, 21
ISBN checking, 3
version
Boost.Checks, 1
ISBN example, 5
Overview, 2
TODO, 6
Version Info, 6
Version Info
C++, 6
Quickbook, 6
version, 6
VISA
Header < boost/checks/visa.hpp >, 21
Rationale, 6
VISAcheck
Header < boost/checks/VISACheck.hpp >, 21
VISAcompute
Header < boost/checks/VISACheck.hpp >, 21