Boost.Tribool

Douglas Gregor <dgregor -at- cs.indiana.edu>

Table of Contents

Introduction

The 3-state boolean library contains a single class, `boost::logic::tribool`, along with support functions and operator overloads that implement 3-state boolean logic.

Tutorial

Basic usage

The `tribool` class acts like the built-in `bool` type, but for 3-state boolean logic. The three states are `true`, `false`, and `indeterminate`, where the first two states are equivalent to those of the C++ `bool` type and the last state represents an unknown boolean value (that may be `true` or `false`, we don't know).

The `tribool` class supports conversion from `bool` values and literals along with its own `indeterminate` keyword:

```
tribool b(true);
b = false;
b = indeterminate;
tribool b2(b);
```

`tribool` supports conversions to `bool` for use in conditional statements. The conversion to `bool` will be `true` when the value of the `tribool` is always true, and `false` otherwise. Consequently, the following idiom may be used to determine which of the three states a `tribool` currently holds:

```
tribool b = some_operation();
```

```
if (b) {
  // b is true
}
else if (!b) {
  // b is false
}
else {
  // b is indeterminate
}
```

`tribool` supports the 3-state logic operators `!` (negation), `&&` (AND), and `||` (OR), with `bool` and `tribool` values. For instance:

```
tribool x = some_op();
tribool y = some_other_op();
if (x && y) {
  // both x and y are true
}
else if (!(x && y)) {
  // either x or y is false
}
else {
  // neither x nor y is false, but we don't know that both are true

  if (x || y) {
    // either x or y is true
  }
}
```

Similarly, `tribool` supports 3-state equality comparisons via the operators `==` and `!=`. These operators differ from "normal" equality operators in C++ because they return a `tribool`, because potentially we might not know the result of a comparison (try to compare `true` and `indeterminate`). For instance:

```
tribool x(true);
tribool y(indeterminate);

assert(x == x); // okay, x == x returns true
assert(x == true); // okay, can compare tribools and bools
```

The `indeterminate` keyword (representing the `indeterminate` `tribool` value) doubles as a function to check if the value of a `tribool` is indeterminate, e.g.,

```
tribool x = try_to_do_something_tricky();
if (indeterminate(x)) {
  // value of x is indeterminate
}
else {
  // report success or failure of x
}
```

Renaming the indeterminate state

Users may introduce additional keywords for the indeterminate value in addition to the implementation-supplied `indeterminate` using the `BOOST_TRIBOOL_THIRD_STATE` macro. For instance, the following macro instantiation (at the global scope) will introduce the keyword `maybe` as a synonym for `indeterminate` (also residing in the `boost` namespace):

```
BOOST_TRIBOOL_THIRD_STATE(maybe)
tribool x = maybe;
if (maybe(x)) { /* ... */ }
```

`tribool` input/output

`tribool` objects may be read from and written to streams by including the `boost/logic/tribool_io.hpp` header in a manner very similar to `bool` values. When the `boolalpha` flag is not set on the input/output stream, the integral values 0, 1, and 2 correspond to `tribool` values `false`, `true`, and `indeterminate`, respectively. When `boolalpha` is set on the stream, arbitrary strings can be used to represent the three values, the default being "false", "true", and "indeterminate". For instance:

```
tribool x;
cin >> x; // Type "0", "1", or "2" to get false, true, or indeterminate
cout << boolalpha << x; // Produces "false", "true", or "indeterminate"
```

`tribool` input and output is sensitive to the stream's current locale. The strings associated with false and true values are contained in the standard `std::numpunct` facet, and the string naming the indeterminate type is contained in the `indeterminate_name` facet. To replace the name of the indeterminate state, you need to imbue your stream with a local containing a `indeterminate_name` facet, e.g.:

```
BOOST_TRIBOOL_THIRD_STATE(maybe)
locale global;
locale test_locale(global, new indeterminate_name<char>("maybe"));
cout.imbue(test_locale);
tribool x(maybe);
cout << boolalpha << x << endl; // Prints "maybe"
```

If you C++ standard library implementation does not support locales, `tribool` input/output will still work, but you will be unable to customize the strings printed/parsed when `boolalpha` is set.

Reference

Header <boost/logic/tribool.hpp>

```
BOOST_TRIBOOL_THIRD_STATE(Name)


namespace boost {
  namespace logic {
    class tribool;
    bool indeterminate(tribool, unspecified = unspecified);
    tribool operator!(tribool);
    tribool operator&&(tribool, tribool);
    tribool operator&&(tribool, bool);
    tribool operator&&(bool, tribool);
    tribool operator&&(indeterminate_keyword_t, tribool);
    tribool operator&&(tribool, indeterminate_keyword_t);
    tribool operator||(tribool, tribool);
    tribool operator||(tribool, bool);
    tribool operator||(bool, tribool);
    tribool operator||(indeterminate_keyword_t, tribool);
    tribool operator||(tribool, indeterminate_keyword_t);
    tribool operator==(tribool, tribool);
    tribool operator==(tribool, bool);
    tribool operator==(bool, tribool);
    tribool operator==(indeterminate_keyword_t, tribool);
    tribool operator==(tribool, indeterminate_keyword_t);
    tribool operator!=(tribool, tribool);
    tribool operator!=(tribool, bool);
    tribool operator!=(bool, tribool);
    tribool operator!=(indeterminate_keyword_t, tribool);
    tribool operator!=(tribool, indeterminate_keyword_t);
  }
}
```

Class tribool

Class tribool -- A 3-state boolean type.

```
class tribool {
public:
  // construct/copy/destruct
  tribool();
  tribool(bool);
  tribool(indeterminate_keyword_t);

  // public member functions
  operator safe_bool() const;

  enum boost::logic::tribool::value_t value;
};
```

Description

3-state boolean values are either true, false, or indeterminate.

`tribool` construct/copy/destruct

1.
   ```
   tribool();
   ```

   Construct a new 3-state boolean value with the value 'false'.

   Throws Will not throw.

2.
   ```
   tribool(bool value);
   ```

   Construct a new 3-state boolean value with the given boolean value, which may be `true` or `false`.

   Throws Will not throw.

3.
   ```
   tribool(indeterminate_keyword_t );
   ```

   Construct a new 3-state boolean value with an indeterminate value.

   Throws Will not throw.

`tribool` public member functions

1.
   ```
   operator safe_bool() const;
   ```

Use a 3-state boolean in a boolean context. Will evaluate true in a boolean context only when the 3-state boolean is definitely true.

Returns	true if the 3-state boolean is true, false otherwise
Throws	Will not throw.

Function indeterminate

Function indeterminate -- Keyword and test function for the indeterminate tribool value.

```
bool indeterminate(tribool x, unspecified dummy = unspecified);
```

Description

The `indeterminate` function has a dual role. It's first role is as a unary function that tells whether the tribool value is in the "indeterminate" state. It's second role is as a keyword representing the indeterminate (just like "true" and "false" represent the true and false states). If you do not like the name "indeterminate", and would prefer to use a different name, see the macro `BOOST_TRIBOOL_THIRD_STATE`.

Returns	`x.value == tribool::indeterminate_value`
Throws	Will not throw.

Function operator!

Function operator! -- Computes the logical negation of a tribool.

```
tribool operator!(tribool x);
```

Description

Returns the logical negation of the tribool, according to the table:

!	
false	true
true	false
indeterminate	indeterminate

Throws Will not throw.

Function operator&&

Function operator&& -- Computes the logical conjuction of two tribools.

```
tribool operator&&(tribool x, tribool y);
tribool operator&&(tribool x, bool y);
tribool operator&&(bool x, tribool y);
tribool operator&&(indeterminate_keyword_t , tribool x);
tribool operator&&(tribool x, indeterminate_keyword_t );
```

Description

Returns the result of logically ANDing the two tribool values, according to the following table:

&&	**false**	**true**	**indeterminate**
false	false	false	false
true	false	true	indeterminate
indeterminate	false	indeterminate	indeterminate

Throws Will not throw.

Function operator||

Function operator|| -- Computes the logical disjunction of two tribools.

```
tribool operator||(tribool x, tribool y);
tribool operator||(tribool x, bool y);
tribool operator||(bool x, tribool y);
tribool operator||(indeterminate_keyword_t , tribool x);
tribool operator||(tribool x, indeterminate_keyword_t );
```

Description

Returns the result of logically ORing the two tribool values, according to the following table:

||	**false**	**true**	**indeterminate**
false	false	true	indeterminate
true	true	true	true
indeterminate	indeterminate	true	indeterminate

Throws Will not throw.

Function operator==

Function operator== -- Compare tribools for equality.

```
tribool operator==(tribool x, tribool y);
tribool operator==(tribool x, bool y);
tribool operator==(bool x, tribool y);
tribool operator==(indeterminate_keyword_t , tribool x);
tribool operator==(tribool x, indeterminate_keyword_t );
```

Description

Returns the result of comparing two tribool values, according to the following table:

==	**false**	**true**	**indeterminate**
false	true	false	indeterminate
true	false	true	indeterminate
indeterminate	indeterminate	indeterminate	indeterminate

Throws Will not throw.

Function operator!=

Function operator!= -- Compare tribools for inequality.

```
tribool operator!=(tribool x, tribool y);
tribool operator!=(tribool x, bool y);
tribool operator!=(bool x, tribool y);
tribool operator!=(indeterminate_keyword_t , tribool x);
tribool operator!=(tribool x, indeterminate_keyword_t );
```

Description

Returns the result of comparing two tribool values for inequality, according to the following table:

!=	**false**	**true**	**indeterminate**
false	false	true	indeterminate
true	true	false	indeterminate
indeterminate	indeterminate	indeterminate	indeterminate

Throws Will not throw.

Macro BOOST_TRIBOOL_THIRD_STATE

Macro BOOST_TRIBOOL_THIRD_STATE -- Declare a new name for the third state of a tribool.

```
BOOST_TRIBOOL_THIRD_STATE(Name)
```

Description

Use this macro to declare a new name for the third state of a tribool. This state can have any number of new names (in addition to `indeterminate`), all of which will be equivalent. The new name will be placed in the namespace in which the macro is expanded.

Example: BOOST_TRIBOOL_THIRD_STATE(true_or_false)

tribool x(true_or_false); // potentially set x if (true_or_false(x)) { // don't know what x is }

Header <boost/logic/tribool_fwd.hpp>

```
namespace boost {
  namespace logic {
  }
}
```

Header <boost/logic/tribool_io.hpp>

```
namespace boost {
  namespace logic {
    template<typename CharT> class indeterminate_name;
    template<typename T>
      std::basic_string< T > get_default_indeterminate_name();

    // Returns the character string "indeterminate".
    template<>
      std::basic_string< char > get_default_indeterminate_name<char >();

    // Returns the wide character string L"indeterminate".
    template<>
      std::basic_string< wchar_t > get_default_indeterminate_name<wchar_t >();
    template<typename CharT, typename Traits>
      std::basic_ostream< CharT, Traits > &
      operator<<(std::basic_ostream< CharT, Traits > &, tribool);
    template<typename CharT, typename Traits>
      std::basic_ostream< CharT, Traits > &
      operator<<(std::basic_ostream< CharT, Traits > &, unspecified);
    template<typename CharT, typename Traits>
      std::basic_istream< CharT, Traits > &
      operator>>(std::basic_istream< CharT, Traits > &, tribool &);
  }
}
```

Class template indeterminate_name

Class template indeterminate_name -- A locale facet specifying the name of the indeterminate value of a tribool.

```
template<typename CharT> 
class indeterminate_name {
public:
  // types
  typedef CharT                      char_type;  
  typedef std::basic_string< CharT > string_type;

  // construct/copy/destruct
  indeterminate_name();
  indeterminate_name(const string_type &);

  // public member functions
  string_type name() const;

  static std::locale::id id;
};
```

Description

The facet is used to perform I/O on tribool values when `std::boolalpha` has been specified. This class template is only available if the C++ standard library implementation supports locales.

`indeterminate_name` construct/copy/destruct

1.
   ```
   indeterminate_name();
   ```

2.
   ```
   indeterminate_name(const string_type & name);
   ```

`indeterminate_name` public member functions

1.
   ```
   string_type name() const;
   ```

Function template get_default_indeterminate_name

Function template get_default_indeterminate_name -- Returns a string containing the default name for the indeterminate value of a tribool with the given character type T.

```
template<typename T> std::basic_string< T > get_default_indeterminate_name();
```

Description

This routine is used by the input and output streaming operators for tribool when there is no locale support or the stream's locale does not contain the indeterminate_name facet.

Function template operator<<

Function template operator<< -- Writes the value of a tribool to a stream.

```
template<typename CharT, typename Traits>
  std::basic_ostream< CharT, Traits > &
  operator<<(std::basic_ostream< CharT, Traits > & out, tribool x);
```

Description

When the value of `x` is either `true` or `false` , this routine is semantically equivalent to:

```
 out << static_cast<bool>(x);
```

When `x` has an indeterminate value, it outputs either the integer value 2 (if `(out.flags() & std::ios_base::boolalpha) == 0`) or the name of the indeterminate value. The name of the indeterminate value comes from the indeterminate_name facet (if it is defined in the output stream's locale), or from the get_default_indeterminate_name function (if it is not defined in the locale or if the C++ standard library implementation does not support locales).

Returns	`out`

Function template operator<<

Function template operator<< -- Writes the indeterminate tribool value to a stream.

```
template<typename CharT, typename Traits>
  std::basic_ostream< CharT, Traits > &
  operator<<(std::basic_ostream< CharT, Traits > & out, unspecified );
```

Description

This routine outputs either the integer value 2 (if `(out.flags() & std::ios_base::boolalpha) == 0`) or the name of the indeterminate value. The name of the indeterminate value comes from the indeterminate_name facet (if it is defined in the output stream's locale), or from the get_default_indeterminate_name function (if it is not defined in the locale or if the C++ standard library implementation does not support locales).

Returns	`out`

Function template operator>>

Function template operator>> -- Reads a tribool value from a stream.

```
template<typename CharT, typename Traits>
  std::basic_istream< CharT, Traits > &
  operator>>(std::basic_istream< CharT, Traits > & in, tribool & x);
```

Description

When `(out.flags() & std::ios_base::boolalpha) == 0` , this function reads a `long` value from the input stream `in` and converts that value to a tribool. If that value is 0, `x` becomes `false` ; if it is 1, `x` becomes `true` ; if it is 2, `becomesindetermine` ; otherwise, the operation fails (and the fail bit is set on the input stream `in`).

When `(out.flags() & std::ios_base::boolalpha) != 0` , this function first determines the names of the false, true, and indeterminate values. The false and true names are extracted from the `std::numpunct` facet of the input stream's locale (if the C++ standard library implementation supports locales), or from the `default_false_name` and `default_true_name` functions (if there is no locale support). The indeterminate name is extracted from the appropriate `indeterminate_name` facet (if it is available in the input stream's locale), or from the `get_default_indeterminate_name` function (if the C++ standard library implementation does not support locales, or the `indeterminate_name` facet is not specified for this locale object). The input is then matched to each of these names, and the tribool `x` is assigned the value corresponding to the longest name that matched. If no name is matched or all names are empty, the operation fails (and the fail bit is set on the input stream `in`).

Returns `in`

Testsuite

Acceptance tests

Test	Type	Description
tribool_test.cpp	run	Test all features of the `boost::logic::tribool` class.
tribool_rename_test.cpp	run	Test the use of the `BOOST_TRIBOOL_THIRD_STATE` macro.
tribool_io_test.cpp	run	Test tribool input/output.