Chapter 27. Boost.Tribool

The library Boost.Tribool provides the class `boost::logic::tribool`, which is similar to `bool`. However, while `bool` can distinguish two states, `boost::logic::tribool` handles three.

To use `boost::logic::tribool`, include the header file `boost/logic/tribool.hpp`.

Example 27.1. Three states of `boost::logic::tribool`

```
#include <boost/logic/tribool.hpp>
#include <iostream>

using namespace boost::logic;

int main()
{
  tribool b;
  std::cout << std::boolalpha << b << '\n';

  b = true;
  b = false;
  b = indeterminate;
  if (b)
    ;
  else if (!b)
    ;
  else
    std::cout << "indeterminate\n";
}
```

A variable of type `boost::logic::tribool` can be set to `true`, `false`, or `indeterminate`. The default constructor initializes the variable to `false`. That's why Example 27.1 writes `false` first.

The `if` statement in Example 27.1 illustrates how to evaluate **b** correctly. You have to check for `true` and `false` explicitly. If the variable is set to `indeterminate`, as in the example, the `else` block will be executed.

Boost.Tribool also provides the function `boost::logic::indeterminate()`. If you pass a variable of type `boost::logic::tribool` that is set to `indeterminate`, this function will return `true`. If the variable is set to `true` or `false`, it will return `false`.

Example 27.2. Logical operators with `boost::logic::tribool`

```
#include <boost/logic/tribool.hpp>
#include <boost/logic/tribool_io.hpp>
#include <iostream>

using namespace boost::logic;

int main()
{
  std::cout.setf(std::ios::boolalpha);

  tribool b1 = true;
  std::cout << (b1 || indeterminate) << '\n';
  std::cout << (b1 && indeterminate) << '\n';

  tribool b2 = false;
  std::cout << (b2 || indeterminate) << '\n';
  std::cout << (b2 && indeterminate) << '\n';

  tribool b3 = indeterminate;
  std::cout << (b3 || b3) << '\n';
  std::cout << (b3 && b3) << '\n';
}
```

You can use logical operators with variables of type `boost::logic::tribool`, just as you can with variables of type `bool`. In fact, this is the only way to process variables of type `boost::logic::tribool` because the class doesn’t provide any member functions.

Example 27.2 returns `true` for `b1 || indeterminate`, `false` for `b2 && indeterminate`, and `indeterminate` in all other cases. If you look at the operations and their results, you will notice that `boost::logic::tribool` behaves as one would expect intuitively. The documentation on Boost.Tribool also contains tables that show which operations lead to which results.

Example 27.2 also illustrates how the values `true`, `false`, and `indeterminate` are written to standard output with variables of type `boost::logic::tribool`. The header file `boost/logic/tribool_io.hpp` must be included and the flag **`std::ios::boolalpha`** must be set for standard output.

Boost.Tribool also provides the macro `BOOST_TRIBOOL_THIRD_STATE`, which lets you substitute another value for `indeterminate`. For example, you could use `dontknow` instead of `indeterminate`.