Chapter 42. Boost.Ref

The library Boost.Ref provides two functions, `boost::ref()` and `boost::cref()`, in the header file `boost/ref.hpp`. They are useful if you use, for example, `std::bind()` for a function which expects parameters by reference. Because `std::bind()` takes parameters by value, you have to deal with references explicitly.

Boost.Ref was added to the standard library in C++11, where you will find the functions `std::ref()` and `std::cref()` in the header file `functional`.

Example 42.1. Using `boost::ref()`

```cpp
#include <boost/ref.hpp>
#include <vector>
#include <algorithm>
#include <functional>
#include <iostream>

void print(std::ostream &os, int i)
{
  os << i << std::endl;
}

int main()
{
  std::vector<int> v{1, 3, 2};
  std::for_each(v.begin(), v.end(),
    std::bind(print, boost::ref(std::cout), std::placeholders::_1));
}
```

In Example 42.1, the function `print()` is passed to `std::for_each()` to write the numbers in **v** to an output stream. Because `print()` expects two parameters – an output stream and the number to be written – `std::bind()` is used. The first parameter passed to `print()` through `std::bind()` is **`std::cout`**. However, `print()` expects a reference to an output stream, while `std::bind()` passes parameters by value. Therefore, `boost::ref()` is used to wrap **`std::cout`**. `boost::ref()` returns a proxy object that contains a reference to the object passed to it. This makes it possible to pass a reference to **`std::cout`** even though `std::bind()` takes all parameters by value.

The function template `boost::cref()` lets you pass a `const` reference.