Chapter 52. Boost.Foreach

Boost.Foreach provides a macro that simulates the range-based `for` loop from C++11. You can use the macro `BOOST_FOREACH`, defined in `boost/foreach.hpp`, to iterate over a sequence without using iterators. If your development environment supports C++11, you can ignore Boost.Foreach.

Example 52.1. Using `BOOST_FOREACH` and `BOOST_REVERSE_FOREACH`

```
#include <boost/foreach.hpp>
#include <array>
#include <iostream>

int main()
{
  std::array<int, 4> a{{0, 1, 2, 3}};

  BOOST_FOREACH(int &i, a)
    i *= i;

  BOOST_REVERSE_FOREACH(int i, a)
  {
    std::cout << i << '\n';
  }
}
```

`BOOST_FOREACH` expects two parameters. The first parameter is a variable or reference, and the second is a sequence. The type of the first parameter needs to match the type of the elements in the sequence.

Anything offering iterators, such as containers from the standard library, classifies as a sequence. Boost.Foreach uses Boost.Range instead of directly accessing the member functions `begin()` and `end()`. However, because Boost.Range is based on iterators, anything providing iterators is compatible with `BOOST_FOREACH`.

Example 52.1 iterates over an array of type `std::array` with `BOOST_FOREACH`. The first parameter passed is a reference so that you can both read and modify the elements in the array. In Example 52.1, the first loop multiplies each number by itself.

The second loop uses the macro `BOOST_REVERSE_FOREACH`, which works the same as `BOOST_FOREACH`, but iterates backwards over a sequence. The loop writes the numbers 9, 4, 1, and 0 in that order to the standard output stream.

As usual, curly brackets can be omitted if the block only consists of one statement.

Please note that you should not use operations that invalidate the iterator inside the loop. For example, elements should not be added or removed while iterating over a vector. `BOOST_FOREACH` and `BOOST_REVERSE_FOREACH` require iterators to be valid throughout the whole iteration.