Boost.Range

Sebastian Redl

C++Now! 2012 Library in a Week

Overview

- Concepts
- Algorithms
- Adapters

Concepts

- Range is a single object representing a half-open iterator range
  - boost::begin and boost::end for access
  - Refinements on traversal, as for iterators
  - Single Pass, Forward, Bidi, Random Access

Algorithms

- All standard (and a few extension) algorithms
- Range version takes a Range instead of a pair of iterators

Algorithms

```
#include <vector>
#include <boost/range/algorithm.hpp>
void f() {
    std::vector<int> v = {1, 2, 3, 4, 5};
    boost::range::random_shuffle(v);
    boost::range::sort(v);
}
```

Adaptors

- Lazy modification of range behavior
- transform, filter, join, ...

Adaptors

```
#include <vector>
#include <iostream>
#include <boost/range/algorithm.hpp>
#include <boost/range/adaptors.hpp>
void f() {
    std::vector<int> v = {1, 2, 3, 4, 5};
    boost::range::for_each(
        v | boost::adaptors::filtered(is_even),
        [](int i) { std::cout << i << '\n'; }
    );
}
```

Adaptors

```
// includes omitted
namespace rga = boost::adaptors;
void f() {
    std::vector<int> v = {1, 2, 3, 4, 5};
    auto with_index = v | rga::indexed;
    for (auto it = boost::begin(with_index);
          it != boost::end(with_index); ++it)
       cout << “At “ << it.index()
          “:“ << *it << ‘\n’;
    }
}
```

Adaptors

```
// includes omitted
template <typename Range1, typename Range2>
using zip_range = boost::iterator_range<
    boost::zip_iterator<boost::tuple<
        typename boost::range_iterator<Range1>::type,
        typename boost::range_iterator<Range2>::type
>>>;
```

Adaptors ctd.

```
template <typename Range1, typename Range2>
zip_range<Range1, Range2> zip(const Range1& r1,
                              const Range2 &r2) {
    return zip_range<Range1, Range2>(
        boost::make_zip_iterator(
            boost::make_tuple(boost::begin(r1),
                              boost::begin(r2))),
        boost::make_zip_iterator(
            boost::make_tuple(boost::end(r1),
                              boost::end(r2))));
}
```

Adaptors ctd.

```
void f() {
    std::vector<int> v = {1, 2, 3, 4, 5};
    for (auto p : zip(v, irange(size_t(0), v.size()))) {
        cout << “At “ << p.get<1>()
            “:“ << p.get<0>() << ‘\n’;
    }
}
```

Questions?