Intro to C++20 Ranges

Ann Arbor C++ Meetup
March 27th, 2019
Timothy C. Wright

A little history

- Standard Template Library introduced generic programming in C++ 1994
    - split algorithms, iterators and containers as separate entities
    - Algorithms required two iterators: the start and the end of the container.

A little history

- STL is verbose.

```
std::vector<int> numbers {1,2,3,4,5,6,7,8,9,10};
std::copy_if(numbers.begin(), numbers.end(),
        std::back_inserter(even_numbers), [](int n){return n%2 ==0;});

auto new_end = std::copy_if(numbers.begin(),numbers.end(),
             numbers.begin(), [](int n){return n%2 ==0;});

numbers.erase(new_end, numbers.end());
std::transform(numbers.begin(), numbers.end(), numbers.begin(),
                 [](int n) { return n*2;});
```

A Range

- boost ranges
- Eric Neibler’s range-v3 library
- What is accepted in C++20. <- focus in this talk

Concepts

- Problems with constraining templates
- Poor error messages
- Using SINFAE as the decision maker

enable_if

```
template<typename T, typename std::enable_if_t<std::is_pod_v<T>, T>* = nullptr>
void foo(T& t)
{…}

struct A { int data;};

A a;
foo(a);

std::vector<A> b;
//foo(b); fails, type of b not a POD
```

requires

```
template<typename T>
requires std::is_pod_v<T>
void foo_requires(T& t)
{}

std::vector<A> b;
foo_requires(b);
```

```
/Users/tcw321/ClionProjects/range_proposal_exercises/concepts.cpp:14:6: note:
constraints not satisfied
 void foo_requires(T& t)
      ^~~~~~~~~~~~
/Users/tcw321/ClionProjects/range_proposal_exercises/concepts.cpp:14:6: note:
'is_pod_v<T>' evaluated to false

* Using gcc-8 compiler with -fconcepts
```

Concepts

Iterator Concepts

Readable

Writable

WeaklyIncrementable

Incrementable

Iterator

Sentinel

SizedSentinel

InputIterator

OutputIterator

ForwardIterator

RandomAccessIterator

ContiguousIterator

Concepts

Range

SizedRange

    Determine size in constant time

View

OutputRange

InputRange

ForwardRange

BidirectionalRange

RandomAccessRange

    Defines []

ContiguousRange

    Defines data()

CommonRange

    begin and end iterators are the same type

Range

- Iterator and a Sentinel : different types
- Solves problems with:
  - delimited range
    - must determine end at run time
  - infinite range
- See Eric Neibler’s posts

Range Example

```
std::vector<int> numbers {1,2,3,4,5,6,7,8,9,10};
auto evenNumbers2 = ranges::filter_view(numbers,
                         [](int n){ return n % 2 == 0; });
auto evenNumbers3 = ranges::transform_view(evenNumbers2,
                         [](int n) { return n * 2; });
```

Range View Adaptors

```
std::vector<int> numbers {1,2,3,4,5,6,7,8,9,10};
auto evenNumbers = numbers | ranges::view::filter([](int n){return n % 2 == 0;})
                           | ranges::view::transform([](int n) { return n * 2; });
```

Views

- Constant time copy and move
- Lazy, does not operate until necessary
- Does not work with modifying algorithms like sort.

in-place algorithms

```
std::vector<int> data{3, 2, 4, 5, 14, 6, 7, 8, 9, 1, 10};
ranges::sort(data);
```

range::v3::actions

```
//auto & v3 = action::sort(v);
v |= action::sort | action::reverse;
std::cout << "action sort\n”;
```

```
Not Lazy
Chainable
In Range-v3 library
Not in C++20
```

view reference

```
auto local_data() {
    using namespace std::experimental;
    std::vector<int> data {1,2,3,4,5,6,7,8,9,10,11};
    auto v = data | ranges::view::filter(
                    [](int n) {return n % 2 == 0;});
    return v;
}
```

```
auto data = local_data();
for (auto d : data)
    std::cout << d << '\n';  // get junk, original ref gone
```

Generate infinite seq

```cpp
using namespace std::experimental;
auto v = ranges::view::iota(1) | ranges::view::take(10) |
         ranges::view::filter([](int n) {return n % 2 == 0;});
```

Generate Sequence

```
auto gen_data() {
    using namespace std::experimental;
    auto v = ranges::view::iota(1) | ranges::view::take(10) |
             ranges::view::filter([](int n) {return n % 2 == 0;});
    return v;
}

..

auto v = gen_data();
for (auto d : v)
    std::cout << d << ", “;  // works, generate data lazily
std::cout << '\n';
```

Projection

```
struct Data {
    int x;
    int y;
    int z;
};

int main() {
    std::vector<Data> values {{1,2,9}, {4,5,6}, {7,8,3}, {21, 4, 7}};
    //std::sort( values.begin(), values.end(), [] (auto &a, auto &b)
    //           { return a.z < b.z;});

    using namespace std::experimental;
    ranges::sort(values, ranges::less{}, &Data::z);
}
```

Surprises

```
std::string text = "Let me split this into words";
auto splitText = text | view::split(' ‘) | view::reverse;
// Fails
// view::split(‘ ‘) returns a ForwardRange which can’t be reversed
//

// Again
auto splitText = text | ranges::view::split(pattern);
static_assert(ranges::ForwardRange<decltype(splitText)>);
static_assert(!ranges::BidirectionalRange<decltype(splitText)>);
for (auto x : splitText)
{
  for (auto m : x)
        std::cout << m;
    std::cout << '\n';
}
```

Other views

In C++ 20

all view
filter view
transform view
iota view
take view
join view
empty view
single view
split view
counted view
common view
reverse view

Others in
ranges::v3

drop
drop_exactly
generate
group_by
slice
sliding
stride
tail
take_while
zip

References

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0896r4.pdf

https://www.youtube.com/watch?v=pe05ZWdh0N0 - Mateusz Pusz C++ Concepts and Ranges

https://www.manning.com/books/functional-programming-in-c-plus-plus

https://www.fluentcpp.com/2018/02/09/introduction-ranges-library/

https://github.com/CaseyCarter/cmcstl2

https://github.com/ericniebler/range-v3

http://ericniebler.com/

https://cpplover.blogspot.com/2019/01/projection-powerful-feature-in-c20.html?m=1

https://cpplover.blogspot.com/2019/01/the-overview-of-c20-range-view.html?m=1