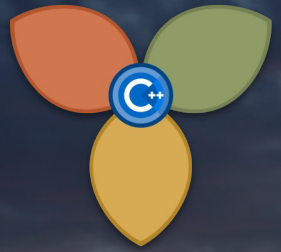


Core C++ 2025

19 Oct. 2025 :: Tel-Aviv

C++20 Ranges for Beginners

By Nathanel Ozeri Green

About me - Nathanel Ozeri Green

Programmer,
Currently working on my own venture (AlphaPulse)

Trainer and Consultant At 

Credit Note

Talk and slides are based on Amir Kirsh’s and Adam Segoli Schubert’s presentation

With their permission, I've adapted and expanded upon their ideas to create this presentation.

std::ranges (since C++20)

An **extension** and **generalization** of the **algorithms** and **iterator** libraries.
Making them more powerful by making them **composable** and **less error-prone**.

https://en.cppreference.com/w/cpp/ranges

Why Ranges?

Motivation

```
std::vector<int> vec{42, 6, 66, 2022, 2024, 1688, -20};
std::sort ( vec.begin(), vec.end() );
std::for_each ( vec.begin(), vec.end(), [](auto val){ cout << val << " "; });
```

https://godbolt.org/z/4eM7s18oP

With std::ranges:

```
std::vector<int> vec{42, 6, 66, 2022, 2024, 1688, -20};
std::ranges::sort ( vec ); // clearer, shorter, simpler
std::ranges::for_each ( vec, [](auto val){ cout << val << " "; });
```

https://godbolt.org/z/1cbjGxh4z

Motivation - Views

```
std::array<int, 7> arr {42, 7, 66, 2023, 2024, 1688, -21};
auto is_odd = [](int n) { return n % 2; };
auto view = arr | std::ranges::views::filter( is_odd ); // not evaluated yet (lazy)
std::ranges::copy(view, std::ostream_iterator<int>{std::cout, ", "});
```

https://godbolt.org/z/88fGn45hd

Motivation - Views

```
std::array<int, 7> arr {42, 7, 66, 2023, 2024, 1688, -21};
auto is_odd = [](int n) { return n % 2; };
auto view = arr | std::ranges::views::filter( is_odd ); // not evaluated yet (lazy)
std::ranges::copy(view, std::ostream_iterator<int>{std::cout, ", "});
```

https://godbolt.org/z/88fGn45hd

Main Components and Mechanics

Main Components

- **range**: collection or sequence that can be iterated over
- **ranges algorithm:** takes ranges and executes eagerly
- **view:** lazily evaluated range that is cheap to copy and assign
- **range adaptor**: makes a view from a range

Mechanics

```
namespace std {
      namespace ranges {...}; // improved algorithms and views
      namespace ranges::views {...}; // range adaptors
      namespace views = ranges::views; // shortcut
}

#include <ranges> // new header for ranges and views
// Ranges stuff also in <algorithm>, <iterator>, <functional>, <numeric>,...
```

Shortcuts for this presentation (but quite common)

```
namespace rng = std::ranges;
namespace rv = std::ranges::views;
```

Range

Range is a C++20 concept that requires begin and end

- Requires **ranges::begin(r)** that returns an **iterator**
- Requires **ranges::end(r)** that returns a **sentinel**
- **sentinel** and **iterator** can be different types

```
template< class R >
concept range = requires( R& r ) {
    ranges::begin(r); // equality-preserving for forward iterators
    ranges::end   (r);
};
```

https://en.cppreference.com/w/cpp/ranges/range

A tiny diversion

What’s a Concept?

- A constraint on template arguments
- Evaluated to a boolean value, at compile time
- May be used for overload resolution

(Added in C++20)

Overloading using a Concept

```
// for types that we can iterate over
void print(const std::ranges::range auto& iterable) {
    for(const auto& v: iterable) {
        cout << v << endl;
    }
}

// for all other types
void print(const auto& t) {
    cout << t << endl;
}
```

Code

Yes, we see here something new! C++20 allows auto on function arguments, making the function a template function, this is called "abbreviated function template"

Back to Ranges

Range

Range is a C++20 concept that requires begin and end

- Requires **ranges::begin(r)** that returns an **iterator**
- Requires **ranges::end(r)** that returns a **sentinel**
- **sentinel** and **iterator** can be different types

```
template< class R >
concept range = requires( R& r ) {
    ranges::begin(r); // equality-preserving for forward iterators
    ranges::end   (r);
};
```

https://en.cppreference.com/w/cpp/ranges/range

Example of Ranges

Standard library (std)

- **Containers: array, vector, map, set, list**
- **Containers: string, string_view**
- **std::filesystem::directory_iterator**
- **std::span**
- **more...**

Many other libraries

Example of Ranges

Standard library (std)

- **Containers: array, vector, map, set, list**
- **Containers: string, string_view**
- **std::filesystem::directory_iterator**
- **std::span**
- **more...**

Many other libraries

Without any required changes in the original classes. They just “happen” to comply to the std::range concept, having a begin and end!

Projections

```
struct info {
   int id = 0;
   std::string name;
};

std::vector<info> vips = {{2, "Cheshire"}, {3, "Sylvester"}, {1, "Garfield"}, {0, "Snoopy"}};
std::ranges::sort(vips, std::ranges::less{}, [](auto const& i) { return i.id; });
```

https://godbolt.org/z/TbePx9MEW

A **Projection** is:

1. Unary callable
2. Modify the view of data (what the algorithm sees)

Projections

```
struct info {
   int id = 0;
   std::string name;
};

std::vector<info> vips = {{2, "Cheshire"}, {3, "Sylvester"}, {1, "Garfield"}, {0, "Snoopy"}};
std::ranges::sort(vips, {}, [](auto const& i) { return i.id; });
```

https://godbolt.org/z/TbePx9MEW

A **Projection** is:

1. Unary callable
2. Modify the view of data (what the algorithm sees)

Views

- Lazily evaluated ranges
    - Iteration is externally driven
- (mostly) non-owning of elements
- Copy, assign, destroy: O(1)
- Can be composed of several processing steps
- Allows infinite range
- Allows pipe syntax

Lazily Reversing

```
std::map<int, std::string> m;
m[1] = "hello";
m[2] = "bybye";
m[3] = "why why?";
m[10] = "no reason...";

for ( auto [k, v] : std::ranges::views::reverse(m) )
{
    cout << k << "->" << v << endl;
}
```

https://godbolt.org/z/MjavnEMnq

```
10->no reason...
3->why why?
2->bybye
1->hello
```

Square an array - Keeping it intact

```
int main() {
   int arr[16]{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
   std::ranges::transform(arr,
                          std::ostream_iterator<int>{std::cout, ", "},
                          [](int i){ return i * i; });
}
```

```
1, 4, 9, 16, 25, 36, 49, 64, 81, 100,
121, 144, 169, 196, 225, 256,
```

https://godbolt.org/z/acPxh9eWh

Print even squares - pipe syntax

```
void print_even_squares(const std::vector<int>& vec) {
   auto is_even = [](auto i){ return !(i % 2); };
   auto square = [](auto i){ return i * i; };

   auto view = vec
       | std::ranges::views::filter(is_even)
       | std::ranges::views::transform(square);

   std::ranges::copy(view, std::ostream_iterator<int>{std::cout, ", "});
}
```

https://godbolt.org/z/8EKn7jW3o

```
4, 16, 36, 64, 100, 144, 196, 256,
```

Additional Example

```
void print(const auto& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
   std::vector container{7, 14, 42, 46, 50};
   print(container);

   print(container | std::views::take(2) | std::views::transform(std::negate{}));
}
```

https://godbolt.org/z/ev5MMdM7Y

```
7, 14, 42, 46, 50,
-7, -14,
```

Views API

Views API

```
void print(const auto& rng) {
   std::cout << rng.size() << " elements\n";
}

int main() {
   std::vector vec{7, 14, 42, 46, 50};
   std::list   lst{7, 14, 42, 46, 50};

   print(vec);
   print(lst);
   print(vec | std::views::take(2));
   print(lst | std::views::take(2));
}
```

https://godbolt.org/z/MxevY4Y58

```
5 elements
5 elements
2 elements
2 elements
```

Views API

```
void print(const auto& rng) {
   std::cout << rng.size() << " elements\n";
}

int main() {
   std::vector vec{7, 14, 42, 46, 50};
   std::list   lst{7, 14, 42, 46, 50};

   print(vec);
   print(lst);
   print(vec | std::views::take(2));
   print(lst | std::views::take(2));
   print(vec | std::views::drop(2));
   print(lst | std::views::drop(2));
}
```

https://godbolt.org/z/chs9T34dz

```
5 elements
5 elements
2 elements
2 elements
?
?
```

Views API

```
void print(const auto& rng) {
   std::cout << rng.size() << " elements\n";
}

int main() {
   std::vector vec{7, 14, 42, 46, 50};
   std::list   lst{7, 14, 42, 46, 50};

   print(vec);
   print(lst);
   print(vec | std::views::take(2));
   print(lst | std::views::take(2));
   print(vec | std::views::drop(2));
   print(lst | std::views::drop(2));
}
```

https://godbolt.org/z/chs9T34dz

```
5 elements
5 elements
2 elements
2 elements
3 elements
3 elements
```

Views API

```
void print(const auto& rng) {
   std::cout << rng.size() << " elements\n";
}

int main() {
   std::vector vec{7, 14, 42, 46, 50};
   std::list   lst{7, 14, 42, 46, 50};

   print(vec);
   print(lst);
   print(vec | std::views::take(2));
   print(lst | std::views::take(2));
   print(vec | std::views::drop(2));
   print(lst | std::views::drop(2));
   print(vec | std::views::filter([](auto n){ return n % 7 == 0;})); // ?
}
```

https://godbolt.org/z/4Kvxa7793

```
5 elements
5 elements
2 elements
2 elements
3 elements
3 elements
?
```

Views API

```
void print(const auto& rng) {
   std::cout << rng.size() << " elements\n";
}

int main() {
   std::vector vec{7, 14, 42, 46, 50};
   std::list   lst{7, 14, 42, 46, 50};

   print(vec);
   print(lst);
   print(vec | std::views::take(2));
   print(lst | std::views::take(2));
   print(vec | std::views::drop(2));
   print(lst | std::views::drop(2));
   print(vec | std::views::filter([](auto n){ return n % 7 == 0;})); // COMPILE ERROR - WHY?
}
```

https://godbolt.org/z/4Kvxa7793

```
5 elements
5 elements
2 elements
2 elements
3 elements
3 elements
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()
lst_view.empty()
lst_view.size()
lst_view[index]
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()        // O(n) - n times ++ on lst.begin()
lst_view.empty()
lst_view.size()
lst_view[index]
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()        // O(n) - n times ++ on lst.begin()
lst_view.empty()        // O(1) - lst.size() <= n
lst_view.size()
lst_view[index]
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()        // O(n) - n times ++ on lst.begin()
lst_view.empty()        // O(1) - lst.size() <= n
lst_view.size()         // O(1) - n >= lst.size() ? 0 : lst.size() - n
lst_view[index]
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()        // O(n) - n times ++ on lst.begin()
lst_view.empty()        // O(1) - lst.size() <= n
lst_view.size()         // O(1) - n >= lst.size() ? 0 : lst.size() - n
lst_view[index]         // O(n) - it's a view on a list
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()        // O(n) - n times ++ on lst.begin()
lst_view.empty()        // O(1) - lst.size() <= n
lst_view.size()         // O(1) - n >= lst.size() ? 0 : lst.size() - n
lst_view[index]         // O(n) - it's a view on a list

auto view = coll | std::views::filter(pred);      // vector or list
view.begin()
view.empty()
view.size()
view[index]
```

Views API

```
std::vector vec{7, 14, 42, 46, 50, ...};
auto vec_view = vec | std::views::drop(n);
vec_view.begin()        // O(1) - vec.begin() + n
vec_view.empty()        // O(1) - vec.size() <= n
vec_view.size()         // O(1) - n >= vec.size() ? 0 : vec.size() - n
vec_view[index]         // O(1) - vec[index + n]

std::list   lst{7, 14, 42, 46, 50, ...};
auto lst_view = lst | std::views::drop(n);
lst_view.begin()        // O(n) - n times ++ on lst.begin()
lst_view.empty()        // O(1) - lst.size() <= n
lst_view.size()         // O(1) - n >= lst.size() ? 0 : lst.size() - n
lst_view[index]         // O(n) - it's a view on a list

auto view = coll | std::views::filter(pred);      // vector or list
view.begin()        // O(n) - until first true - w.c n operations
view.empty()        // O(n) - until first true - w.c n operations
view.size()         // O(n) - pred on all elements
view[index]         // O(n) - until index times true - w.c n operations
```

Member Functions: Complexity

	1st `begin()`	nth `begin()`	`size()`	1st `empty()`	nth `empty()`
std::vector	O(1)	O(1)	O(1)	O(1)	O(1)
std::list	O(1)	O(1)	O(1)	O(1)	O(1)
vec | drop(n)	O(1)	O(1)	O(1)	O(1)	O(1)
lst | drop(n)	O(n)	O(1)	O(1)	O(1)	O(1)
vec | filter(...)	O(n)	O(1)	N / A	O(n)	O(1)
lst | filter(...)	O(n)	O(1)	N / A	O(n)	O(1)
vec | filter(...) | drop(n)	O(n)	O(1)	N / A	O(n)	O(1)
lst | filter(...) | drop(n)	O(n)	O(1)	N / A	O(n)	O(1)

Member Functions: Complexity

	1st `begin()`	nth `begin()`	`size()`	1st `empty()`	nth `empty()`
std::vector	O(1)	O(1)	O(1)	O(1)	O(1)
std::list	O(1)	O(1)	O(1)	O(1)	O(1)
vec | drop(n)	O(1)	O(1)	O(1)	O(1)	O(1)
lst | drop(n)	O(n)	O(1)	O(1)	O(1)	O(1)
vec | filter(.			N / A	O(n)	O(1)
lst | filter(...			N / A	O(n)	O(1)
vec | filter(...) | drop(n)	O(n)	O(1)	N / A	O(n)	O(1)
lst | filter(...) | drop(n)	O(n)	O(1)	N / A	O(n)	O(1)

C++20 Spec: **Amortized O(1)**

But often we call begin only once 😬

Additional compilation errors

What's the issue here?

```
void print(const auto& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  print(vec);
  print(lst);

  print(vec | std::views::drop(3));
  print(lst | std::views::drop(3));  // COMPILATION ERROR - WHY?
}
```

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
COMPILATION ERROR
```

What about this?

```cpp
void print(const auto& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  print(vec);
  print(lst);

  print(vec | std::views::drop(3));
  // print(lst | std::views::drop(3));
  for (int val : lst | std::views::drop(3)) // ?
     std::cout << val << ", ";
}
```

https://godbolt.org/z/qo8f8vWf5

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
?
```

What about this?

```cpp
void print(const auto& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  print(vec);
  print(lst);

  print(vec | std::views::drop(3));
  // print(lst | std::views::drop(3));
  for (int val : lst | std::views::drop(3)) // works!
     std::cout << val << ", ";
}
```

https://godbolt.org/z/qo8f8vWf5

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

What's the issue here?

```
void print(const auto& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  print(vec);
  print(lst);

  print(vec | std::views::drop(3));
  print(lst | std::views::drop(3));  // COMPILATION ERROR - WHY?
}
```

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
COMPILATION ERROR
```

Solution Attempt: forwarding reference

```
void print(auto&& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  print(vec);
  print(lst);

  print(vec | std::views::drop(3));
  print(lst | std::views::drop(3)); // OK
}
```

here

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

https://godbolt.org/z/T1eThe386

Solution Attempt: forwarding reference

```
void print(auto&& rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  print(vec);
  print(lst);

  print(vec | std::views::drop(3));
  print(lst | std::views::drop(3)); // OK
}
```

here

Is this the right solution?

https://godbolt.org/z/T1eThe386

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

Well...

```cpp
int sum;

auto print_and_sum(auto&& rng) {
   std::jthread print_thread{[&rng]{
       std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
       std::cout << '\n';
   }};
   return std::accumulate(rng.begin(), rng.end(), sum);
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  auto sum1 = print_and_sum(vec);
  auto sum2 = print_and_sum(lst);

  auto sum3 = print_and_sum(vec | std::views::drop(3));
  auto sum4 = print_and_sum(lst | std::views::drop(3)); // Undefined Behavior
  std::cout << "sum :" << sum4 << '\n';
}
```

https://godbolt.org/z/4qTbfbjhK

Undefined Behavior:
2 threads calling `begin()` concurrently and it's a write operation.

However...

```
int sum;

auto print_and_sum(const auto& rng) {
   std::jthread print_thread{[&rng]{
       std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
       std::cout << '\n';
   }};
   return std::accumulate (rng.begin(), rng.end(), sum);
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  auto sum1 = print_and_sum(vec);
  auto sum2 = print_and_sum(lst);

  auto sum3 = print_and_sum(vec | std::views::drop(3));
  auto sum4 = print_and_sum(lst | std::views::drop(3)); // COMPILATION ERROR - WHY?
  std::cout << "sum :" << sum4 << '\n';
}
```

Using const auto& prevents this!

https://godbolt.org/z/4qTbfbjhK

However...

```
int sum;

auto print_and_sum(auto&& rng) {
   std::jthread print_thread{[&rng]{
       std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
       std::cout << '\n';
   }};
   return std::accumulate (rng.begin(), rng.end(), sum);
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

  auto sum1 = print_and_sum(vec);
  auto sum2 = print_and_sum(lst);

  auto sum3 = print_and_sum(vec | std::views::drop(3));
  auto sum4 = print_and_sum(lst | std::views::drop(3)); // Undefined Behavior
  std::cout << "sum :" << sum4 << '\n';
}
```

So maybe auto&& ?

Back in a loop!
What else can we do?

https://godbolt.org/z/4qTbfbjhK

Taking by Value?

```
void print(auto rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

   print(vec);
   print(lst);

   print(vec | std::views::drop(3));
   print(lst | std::views::drop(3));
}
```

https://godbolt.org/z/T1eThe386

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

Taking by Value?

```cpp
void print(auto rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
  std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
  std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

   print(vec); // expensive!
   print(lst); // expensive!

   print(vec | std::views::drop(3));
   print(lst | std::views::drop(3));
}
```

https://godbolt.org/z/T1eThe386

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

Taking by Value, as a view?

```cpp
void print(std::ranges::view auto rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
   std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
   std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

   print(vec); // Compile Error
   print(lst); // Compile Error

   print(vec | std::views::drop(3));
   print(lst | std::views::drop(3));
}
```

https://godbolt.org/z/f3dv1fEW7

```
Compile Error
Compile Error

46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

Taking by Value - with views::all

```cpp
void print(std::ranges::view auto rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

int main() {
   std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
   std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

   print(std::views::all(vec)); // OK
   print(std::views::all(lst)); // OK

   print(vec | std::views::drop(3));
   print(lst | std::views::drop(3));
}
```

https://godbolt.org/z/GPcYcWx7c

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

Or, implementing that on the API

```cpp
// for views only
void print(std::ranges::view auto rng) {
   std::ranges::copy(rng, std::ostream_iterator<int>{std::cout, ", "});
   std::cout << '\n';
}

template <typename Container>
void print(const Container& coll)
requires (!std::ranges::view<Container>) { // for non views
   print(std::views::all(coll));
}

int main() {
   std::vector vec{7, 14, 42, 46, 50, 0, 21, 88, 89};
   std::list   lst{7, 14, 42, 46, 50, 0, 21, 88, 89};

   print(vec);
   print(lst);

   print(vec | std::views::drop(3));
   print(lst | std::views::drop(3));
}
```

https://godbolt.org/z/5qKxjazWo

```
7, 14, 42, 46, 50, 0, 21, 88, 89,
7, 14, 42, 46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
46, 50, 0, 21, 88, 89,
```

Sentinels

Sentinels

The need: to allow for “bottomless” ranges (e.g. iota)

In STL end returns an iterator type that actually denotes an “end” (one past the last)

Sentinel marks the end of a range, without actually “pointing” to one past the last

Sentinels

```cpp
template<class InputIt, class T = typename std::iterator_traits<InputIt>::value_type>
constexpr InputIt find(InputIt first, InputIt last, const T& value)
{
    for (; first != last; ++first)
        if (*first == value)
            return first;

    return last;
}
```

```cpp
template<class InputIt, class T>
constexpr InputIt find(InputIt first, unreachable_sentinel_t last, const T& value)
{
    for (;      true      ; ++first)
        if (*first == value)
            return first;

    return last;
}
```

Summary

Summary (1)

Ranges are a great new tool in C++20!

- Many tasks are already implemented there for you, look for them:
  - https://en.cppreference.com
  - https://hackingcpp.com/

- Use them for:
  - Improved algorithms (sometimes)
    - Improved return types
    - Improved efficiency
    - Safer code

Summary (2)

- More readable and elegant code:
  - Work on ranges
  - Projections
  - Pipe syntax

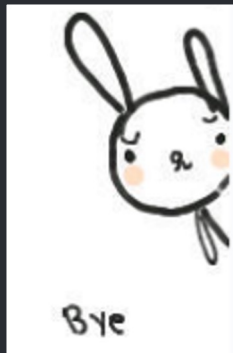
- Pass containers as ranges to hide information and preserve encapsulation.

- Utilize lazy evaluations with views for potentially better performance, lazy evaluation and decreased decoupling (but beware of limitations and lifetime issues!).

Under the Hood and Criticism

Following slides include some additional details and criticism by Nico Josuttis:
C++ Standard Views - Nico Josuttis - ACCU 2023

Any comments or questions before we conclude



Thanks

70