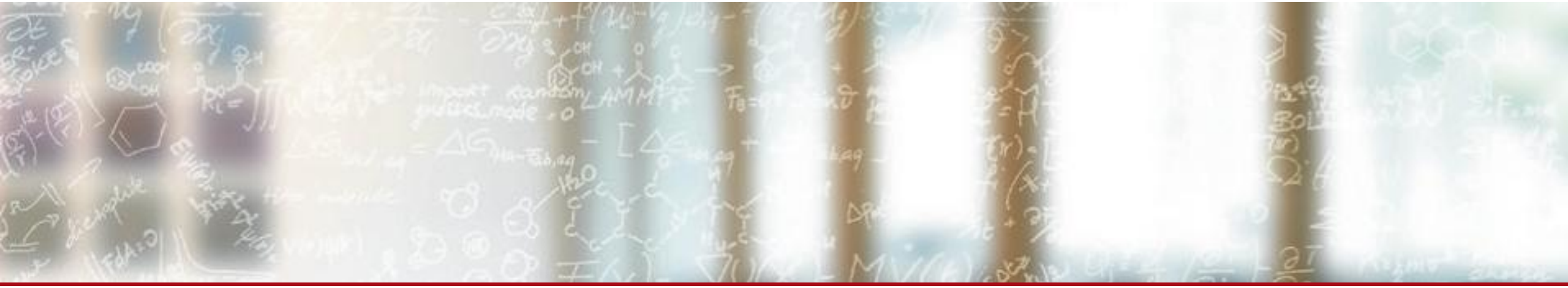
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



C++20 Ranges

Anton Afanasyev, CSCS (afanasyev@cscs.ch)
October, 2021

Resources:

Ideas to improve STL

- Let us call range everything that provides two iterators (in STL sense) -- begin and end;
- Algorithms take ranges where it is possible;
- Let us call views such a ranges that do not have data on their own but refer some other range. Views are cheap to move/copy.

Ideas to improve STL

- Let us call range everything that provides two iterators (in STL sense) -- begin and end;
- Algorithms take ranges where it is possible;
- Let us call views such a ranges that do not have data on their own but refer some other range. Views are cheap to move/copy.
- View adaptors are functors that take a range and return a view. View adaptors are composable.

Old vs. New Interface

```
std::vector v = { 4, 5, 3, 8 };

{
    using namespace std;
    sort(begin(v), end(v));
    sort(rbegin(v), rend(v));
    sort(begin(v) + 1, end(v));
}

{
    using namespace std::ranges;
    sort(v);
    sort(views::reverse(v));
    sort(views::drop(v, 1));
}
```

Pipes

```
std::vector v = { 4, 5, 3, 8 };

{
    using namespace std;
    sort(begin(v), end(v));
    sort(rbegin(v), rend(v));
    sort(begin(v) + 1, end(v));
}

{
    using namespace std::ranges;
    sort(v);
    sort(v | views::reverse);
    sort(v | views::drop(1));
}
```

Niebloids

```
using namespace std::ranges;

std::vector v = { 4, 5, 3, 8 };
sort(v, std::greater<>());
```

Why it compiles?

Niebloids

```
using namespace std::ranges;

std::vector v = { 4, 5, 3, 8 };
sort(v, std::greater<>());
```

Why it compiles?

```
namespace ranges {
    struct sort_t {
        /// overloads for operator()
    };
    inline constexpr sort_t sort = {};
}
```

C++20 Concepts

STL

```
std::list v = { 4, 5, 3, 8 };
std::sort(v.begin(), v.end());
```

```
error: no match for 'operator-' (operand types are 'std::_List_iterator<int>' and 'std::_List_iterator<int>')
```

Ranges

```
std::list v = { 4, 5, 3, 8 };
std::ranges::sort(v);
```

```
error: no match for call to '(const std::ranges::__sort_fn) (std::__cxx11::list<int, std::allocator<int> >&)'
```

Pythagorean Triples. Python

```
import itertools

triples = ((x, y, z)
   for z in itertools.count(1)
       for y in range(1, z + 1)
           for x in range (1, y + 1)
               if z * z == x * x + y * y)

for x in itertools.islice(triples, 10): print (x)
```

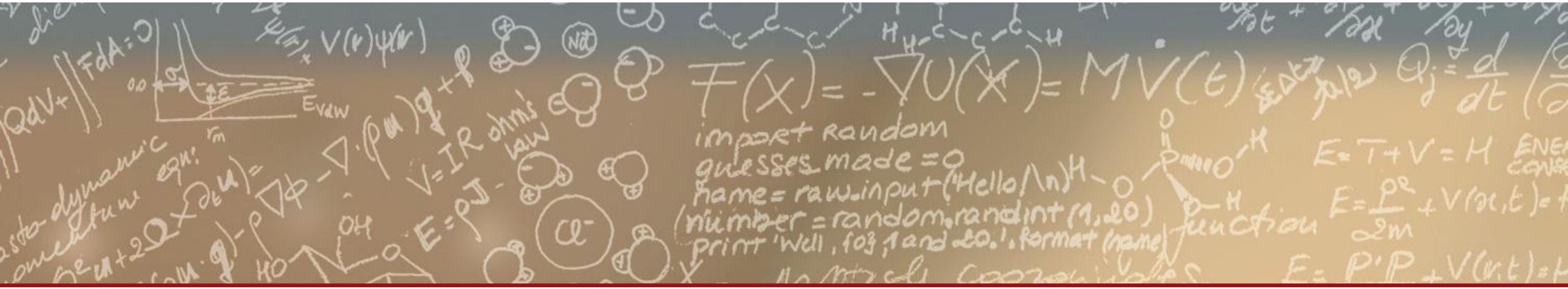
Pythagorean Triples. Ranges

```
auto for_each(auto v, auto f) { return v | views::transform(f) | views::join; }

auto yield_if(bool cond, auto val) { return views::single(val) | views::take(cond); }

auto triples = for_each(views::iota(1), [](auto z) {
   return for_each(views::iota(1, z + 1), [=](auto y) {
       return for_each(views::iota(1, y + 1), [=](auto x) {
           return yield_if(z * z == y * y + x * x, std::array{x, y, z}); });
   });
});

for (auto [x, y, z] : triples | views::take(10))
   std::cout << x << ", " << y << ", " << z << std::endl;
```



Thank you for your attention.