SIMD IN C++20:

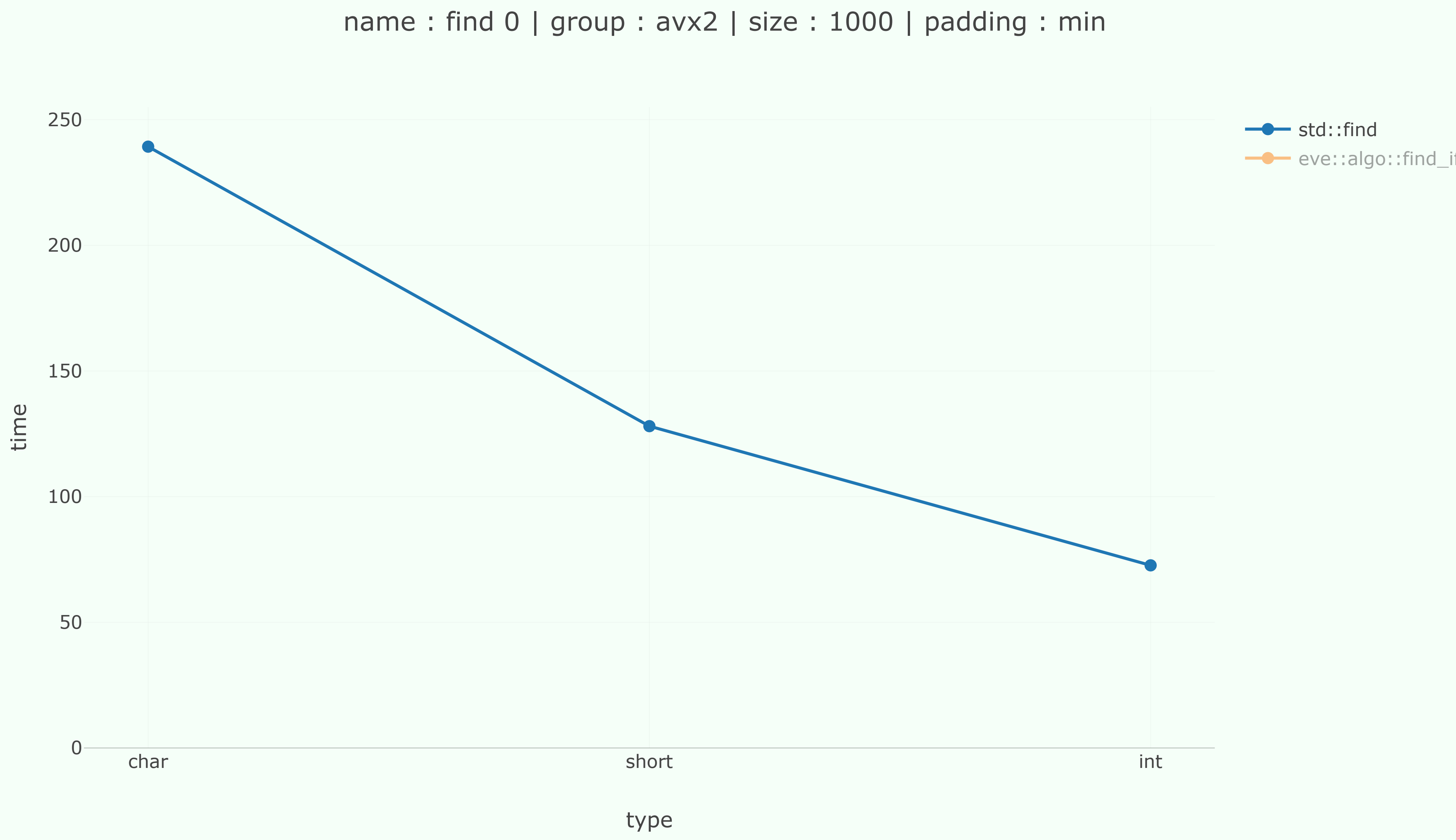
EVE OF A NEW ERA

Joel Falcou & Denis Yaroshevskiy

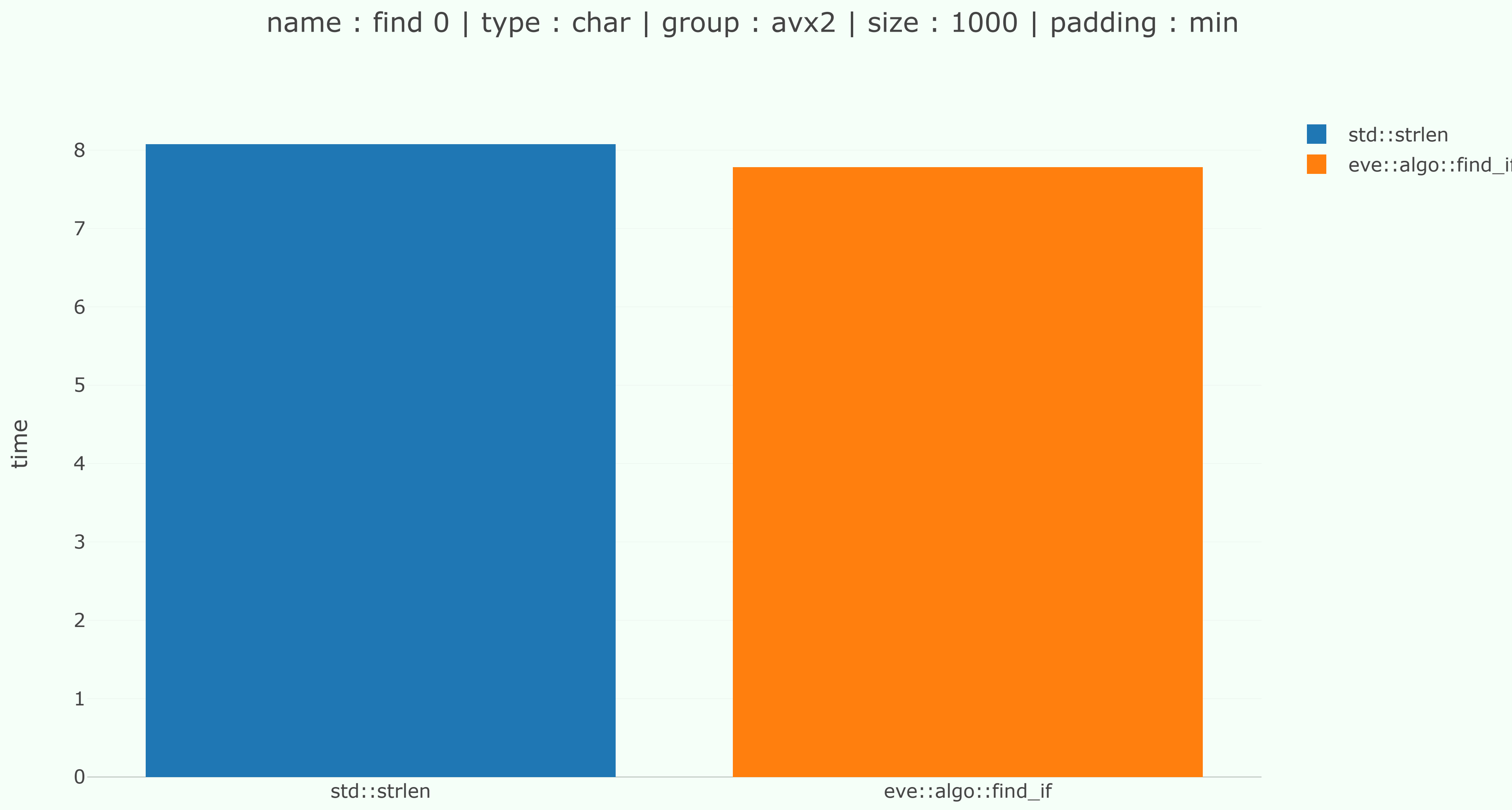
Slides: tinyurl.com/eve-simd-2021

ELEVATOR PITCH

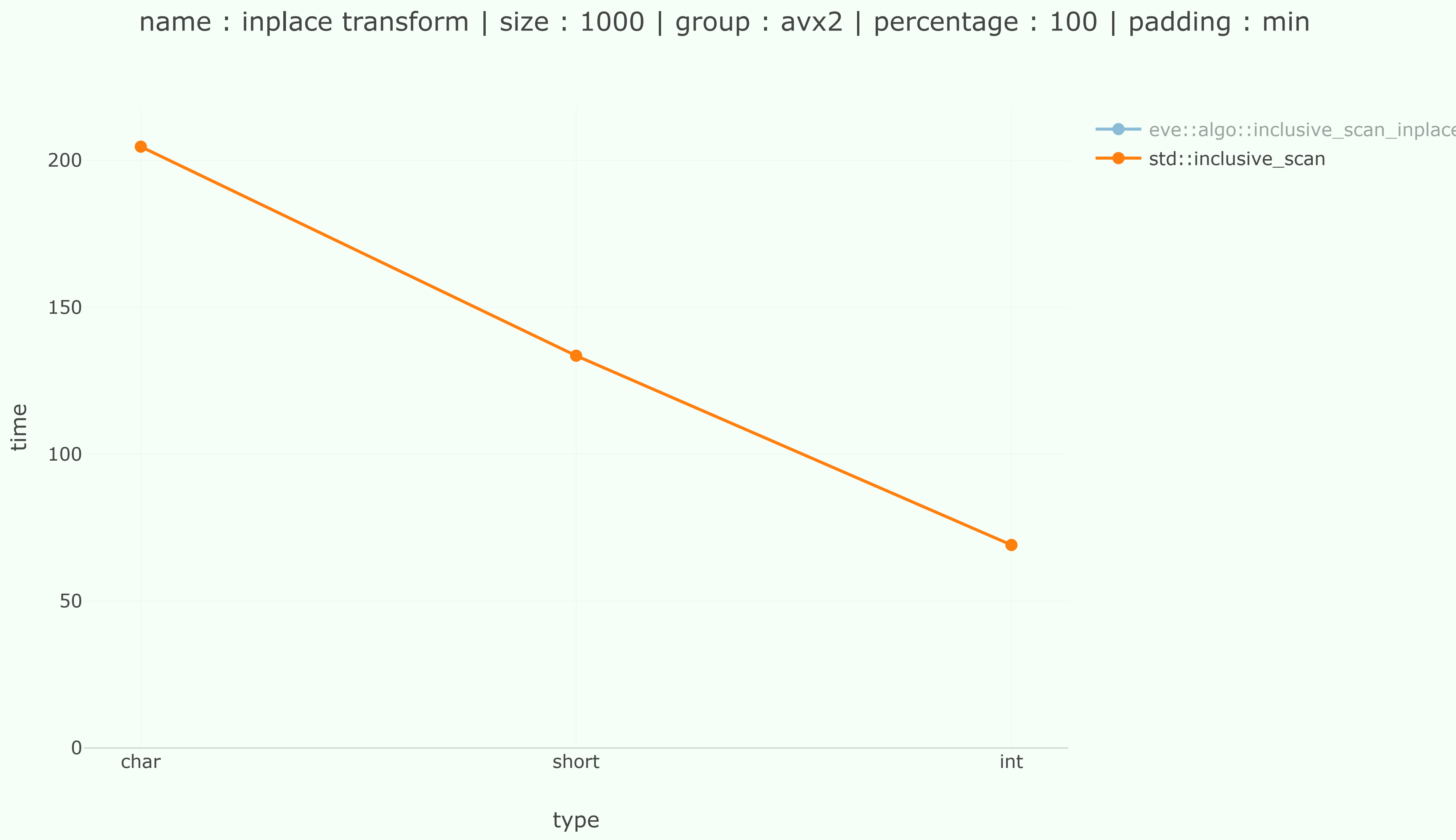
PITCH: FIND (1000 BYTES)



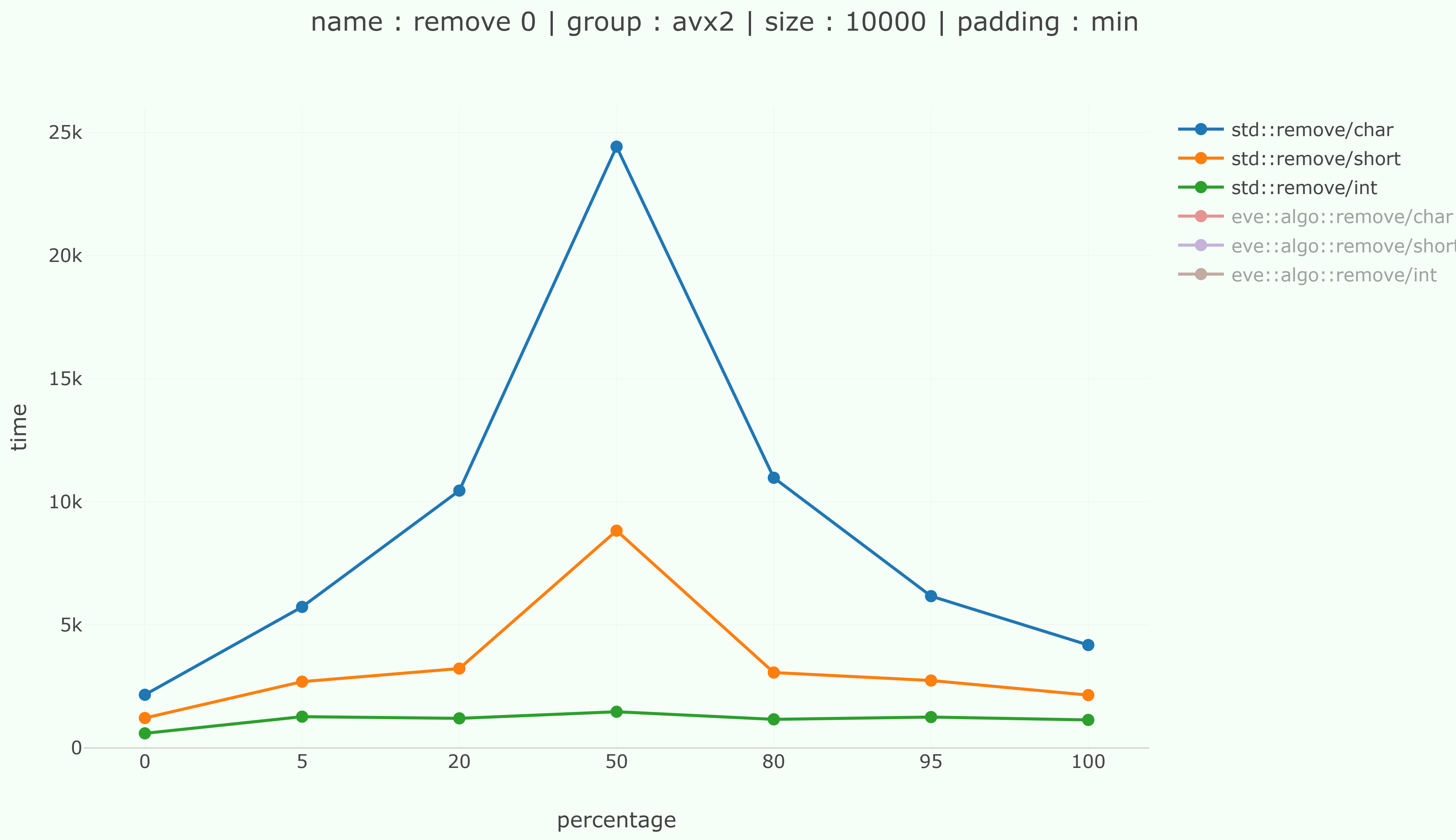
PITCH: EVE VS. GLIBC (1000 BYTES)



PITCH: INCLUSIVE_SCAN (1000 BYTES)

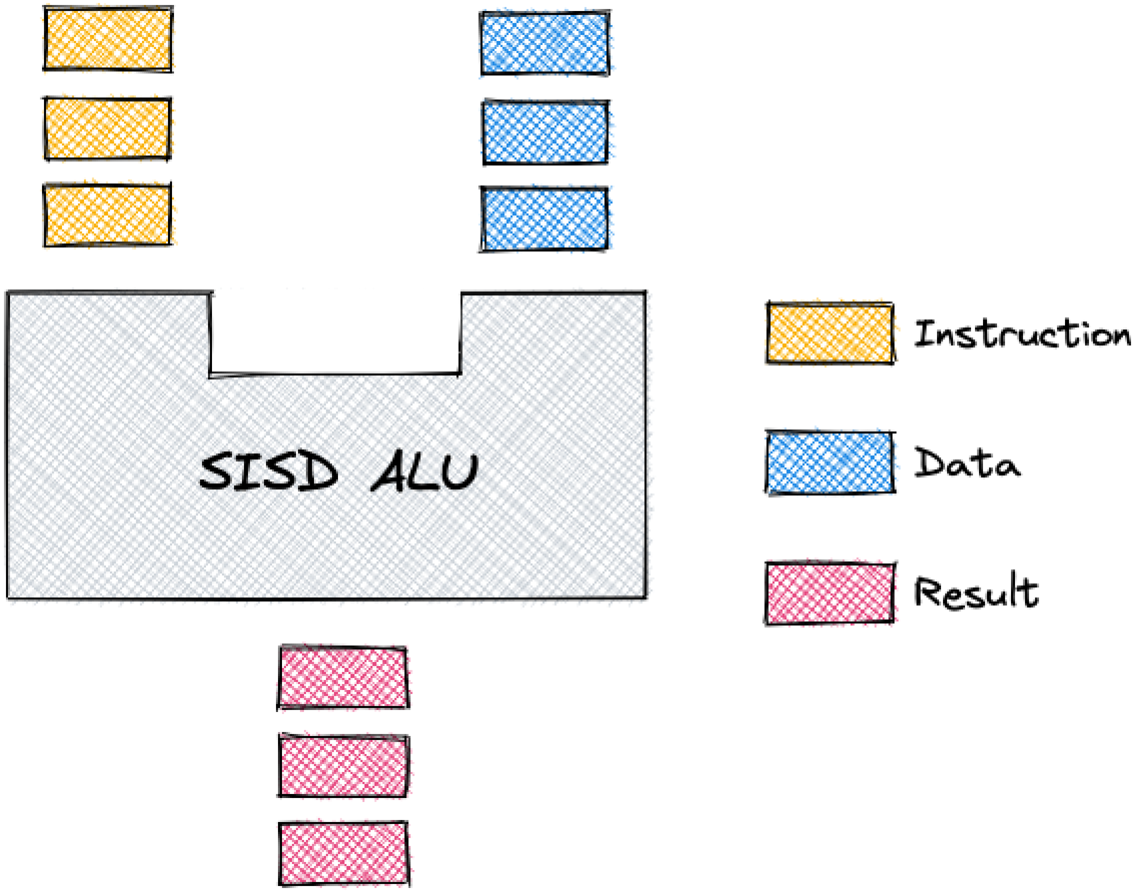


PITCH: REMOVE_IF (10000 BYTES)

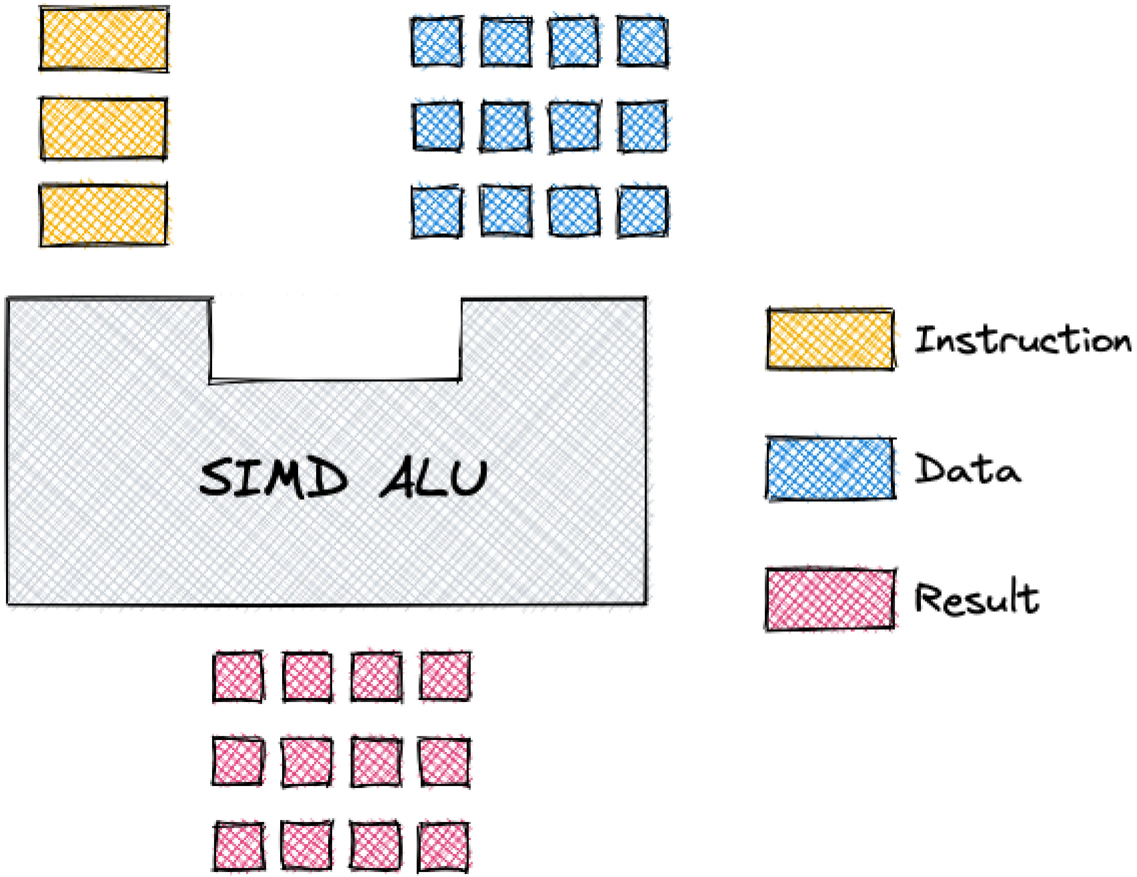


WHAT IS EVE ?

Single Instruction Single Data



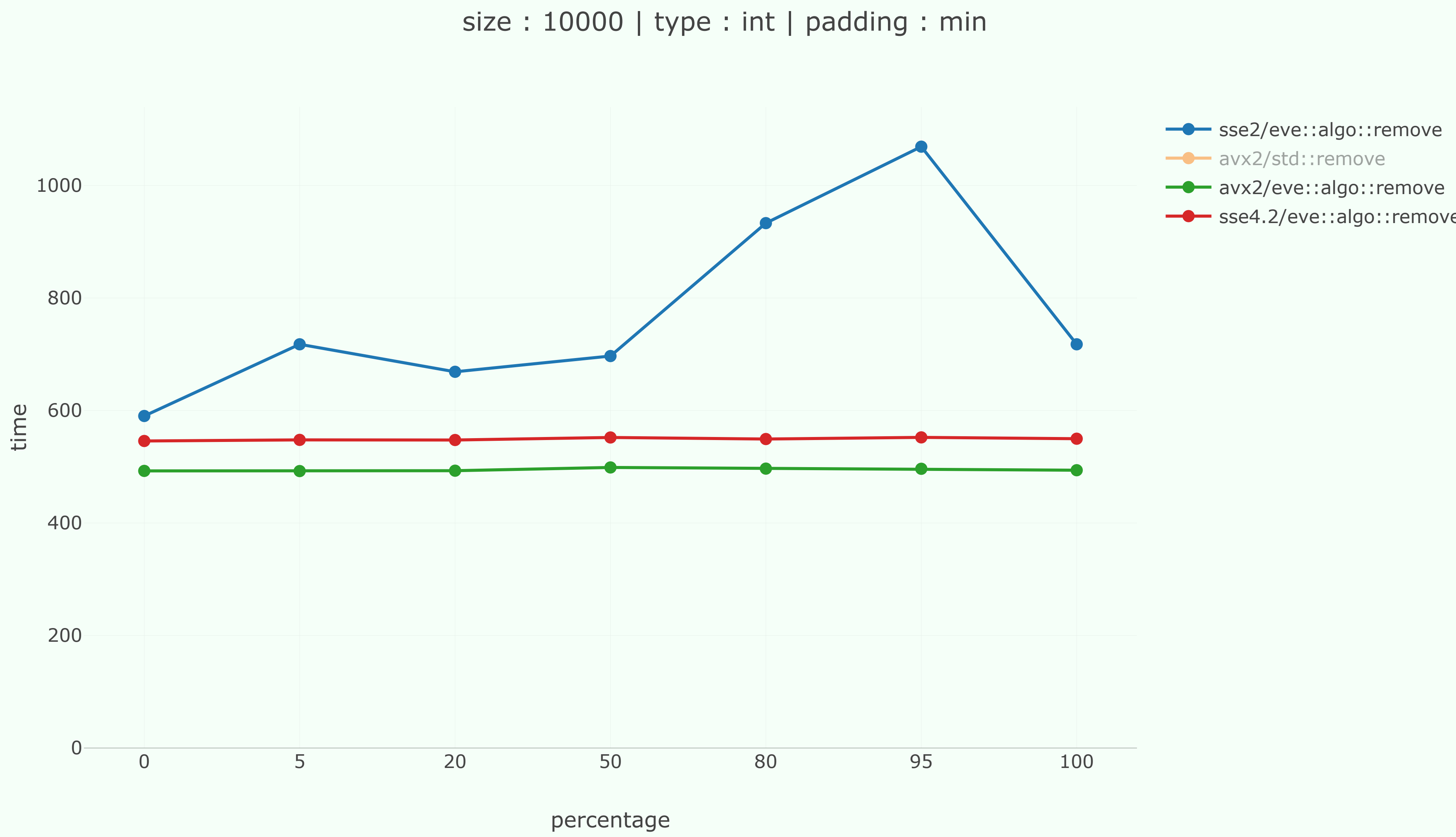
Single Instruction Multiple Data



1001 FLAVORS OF SIMD

- x86
  - 128 bits: SSE2, SSE3, SSSE3, SSE4.1, SSE4.2
  - 256 bits: AVX, AVX2, XOP
  - 512 bits: AVX512 and its myriad of sub-genre
- ARM
  - 128 bits: NEON, ASIMD
- PowerPC
  - 128 bits: Altivec on Power7-9
  - 256 bits: Blue Gene Q QPX

MARCHING ORDERS: REMOVE INT, 10'000



WHAT EXISTS OUT THERE?

WHAT EXISTS OUT THERE?

- Compiler's autovectorization

WHAT EXISTS OUT THERE?

- Compiler's autovectorization
- Pragmas/special compilers

WHAT EXISTS OUT THERE?

- Compiler's autovectorization
- Pragmas/special compilers
- std::execution::unseq, hpx

WHAT EXISTS OUT THERE?

- Compiler's autovectorization
- Pragmas/special compilers
- std::execution::unseq, hpx
- Specialized tools: Halide, simdjson

WHAT EXISTS OUT THERE?

- Compiler's autovectorization
- Pragmas/special compilers
- std::execution::unseq, hpx
- Specialized tools: Halide, simdjson
- write by hand

EVE - EXPRESSIVE VELOCITY ENGINE

EVE - EXPRESSIVE VELOCITY ENGINE

- C++ 20 wrapper around SIMD intrinsincs

EVE - EXPRESSIVE VELOCITY ENGINE

- C++ 20 wrapper around SIMD intrinsincs
  - Library of core types

EVE - EXPRESSIVE VELOCITY ENGINE

- C++ 20 wrapper around SIMD intrinsincs
  - Library of core types
  - Algorithms

EVE - EXPRESSIVE VELOCITY ENGINE

- C++ 20 wrapper around SIMD intrinsincs
  - Library of core types
  - Algorithms
  - 250+ numerical functions

EVE - EXPRESSIVE VELOCITY ENGINE

- C++ 20 wrapper around SIMD intrinsincs
  - Library of core types
  - Algorithms
  - 250+ numerical functions
- Supports all x86 flavors and AARCH64

EVE - EXPRESSIVE VELOCITY ENGINE

- C++ 20 wrapper around SIMD intrinsincs
  - Library of core types
  - Algorithms
  - 250+ numerical functions
- Supports all x86 flavors and AARCH64
- MIT license

SUPPORTED COMPILERS/STANDARDS

SUPPORTED COMPILERS/STANDARDS

- C++20

SUPPORTED COMPILERS/STANDARDS

- C++20
- Clang/GCC

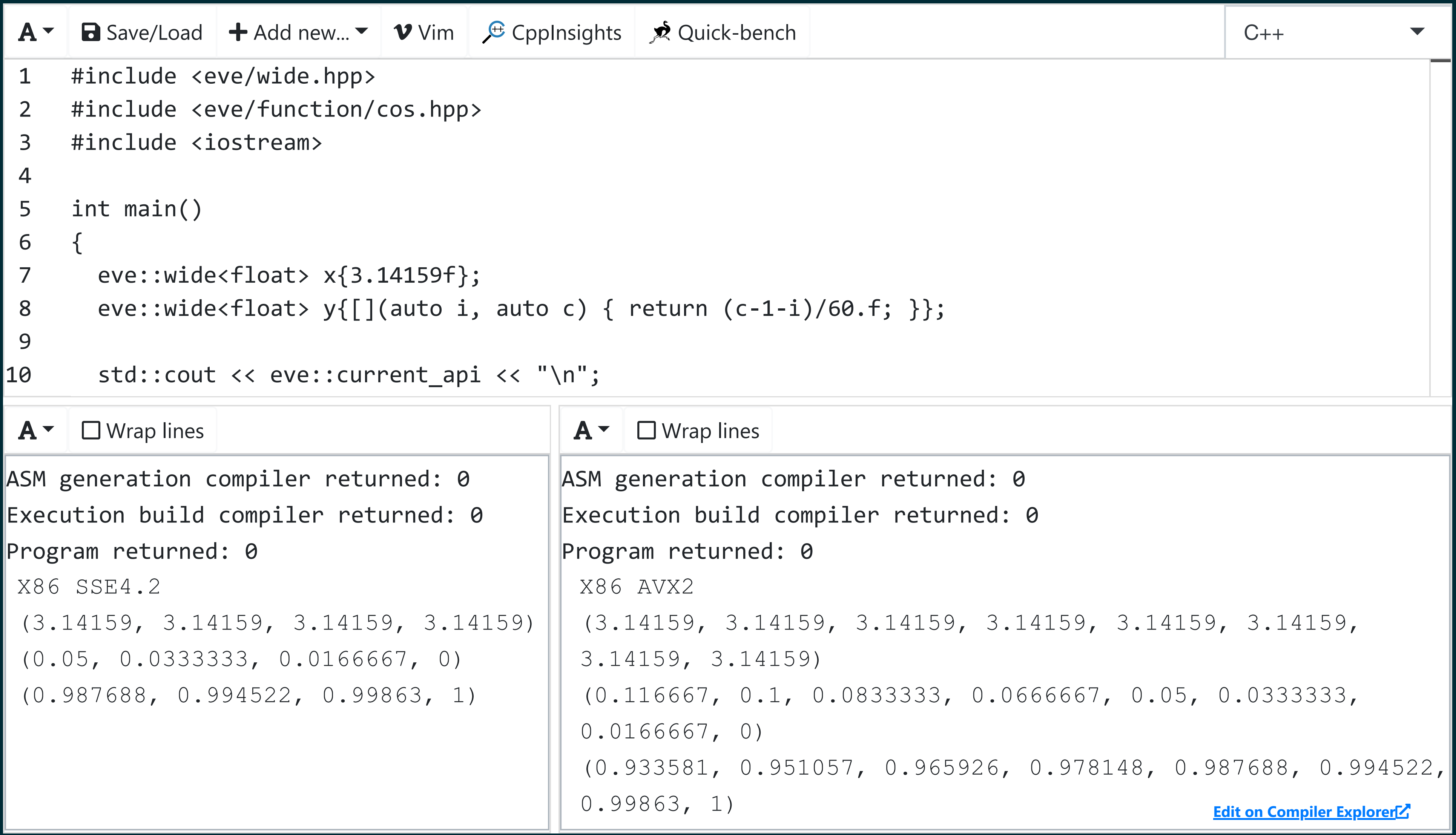
SUPPORTED COMPILERS/STANDARDS

- C++20
- Clang/GCC
- latest until modules

SUPPORTED COMPILERS/STANDARDS

- C++20
- Clang/GCC
- latest until modules
- msvc-clang/msvc latest soon~ish

EVE TYPE WRAPPERS

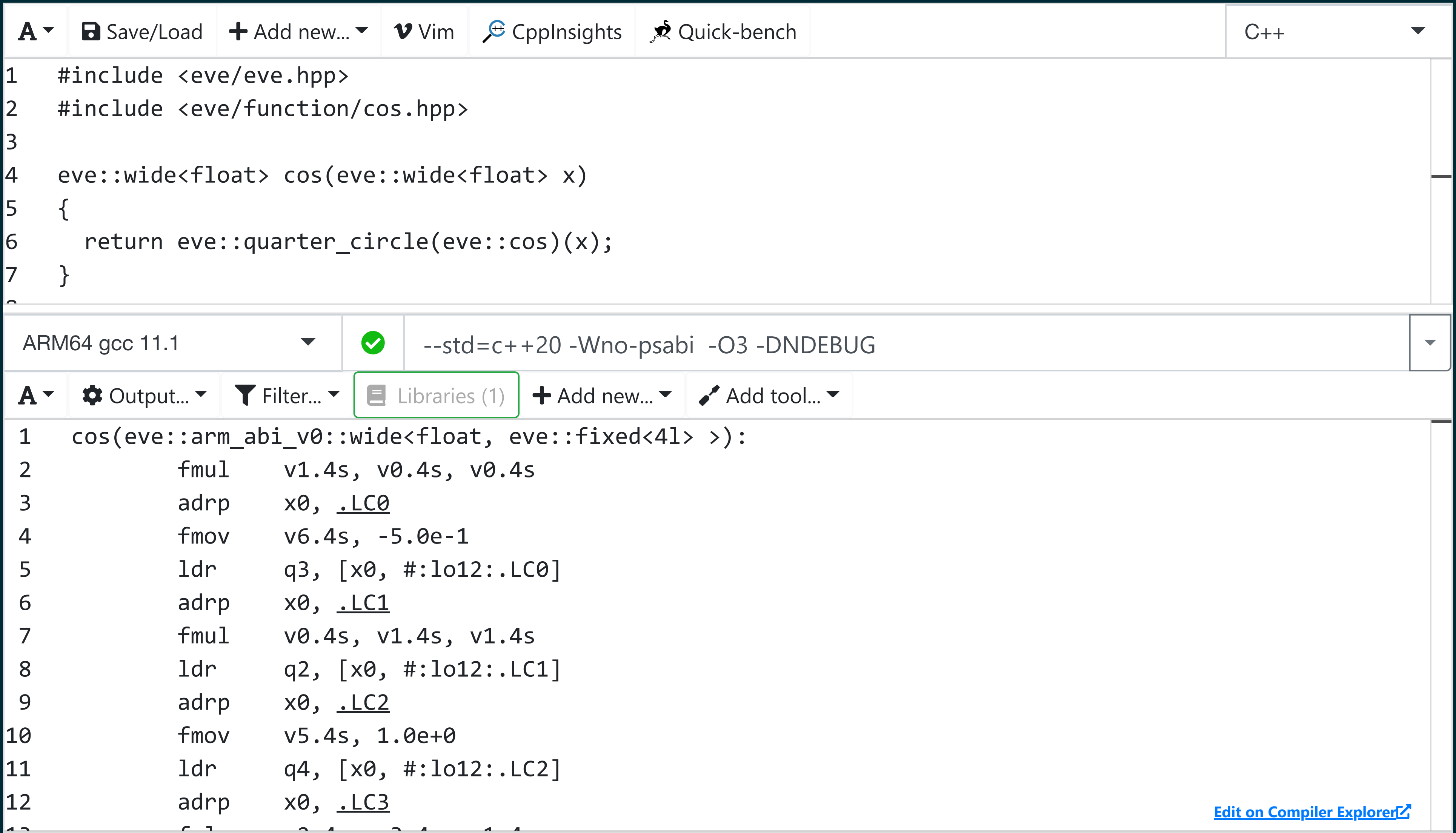


```cpp
#include <eve/wide.hpp>
#include <eve/function/cos.hpp>
#include <iostream>

int main()
{
  eve::wide<float> x{3.14159f};
  eve::wide<float> y{[](auto i, auto c) { return (c-1-i)/60.f; }};

  std::cout << eve::current_api << "\n";
```

```
ASM generation compiler returned: 0
Execution build compiler returned: 0
Program returned: 0
 X86 SSE4.2
 (3.14159, 3.14159, 3.14159, 3.14159)
 (0.05, 0.0333333, 0.0166667, 0)
 (0.987688, 0.994522, 0.99863, 1)
```

```
ASM generation compiler returned: 0
Execution build compiler returned: 0
Program returned: 0
 X86 AVX2
 (3.14159, 3.14159, 3.14159, 3.14159, 3.14159, 3.14159,
 3.14159, 3.14159)
 (0.116667, 0.1, 0.0833333, 0.0666667, 0.05, 0.0333333,
 0.0166667, 0)
 (0.933581, 0.951057, 0.965926, 0.978148, 0.987688, 0.994522,
 0.99863, 1)
```

WHY DECORATORS?



```cpp
#include <eve/eve.hpp>
#include <eve/function/cos.hpp>

eve::wide<float> cos(eve::wide<float> x)
{
  return eve::quarter_circle(eve::cos)(x);
}
```

ARM64 gcc 11.1

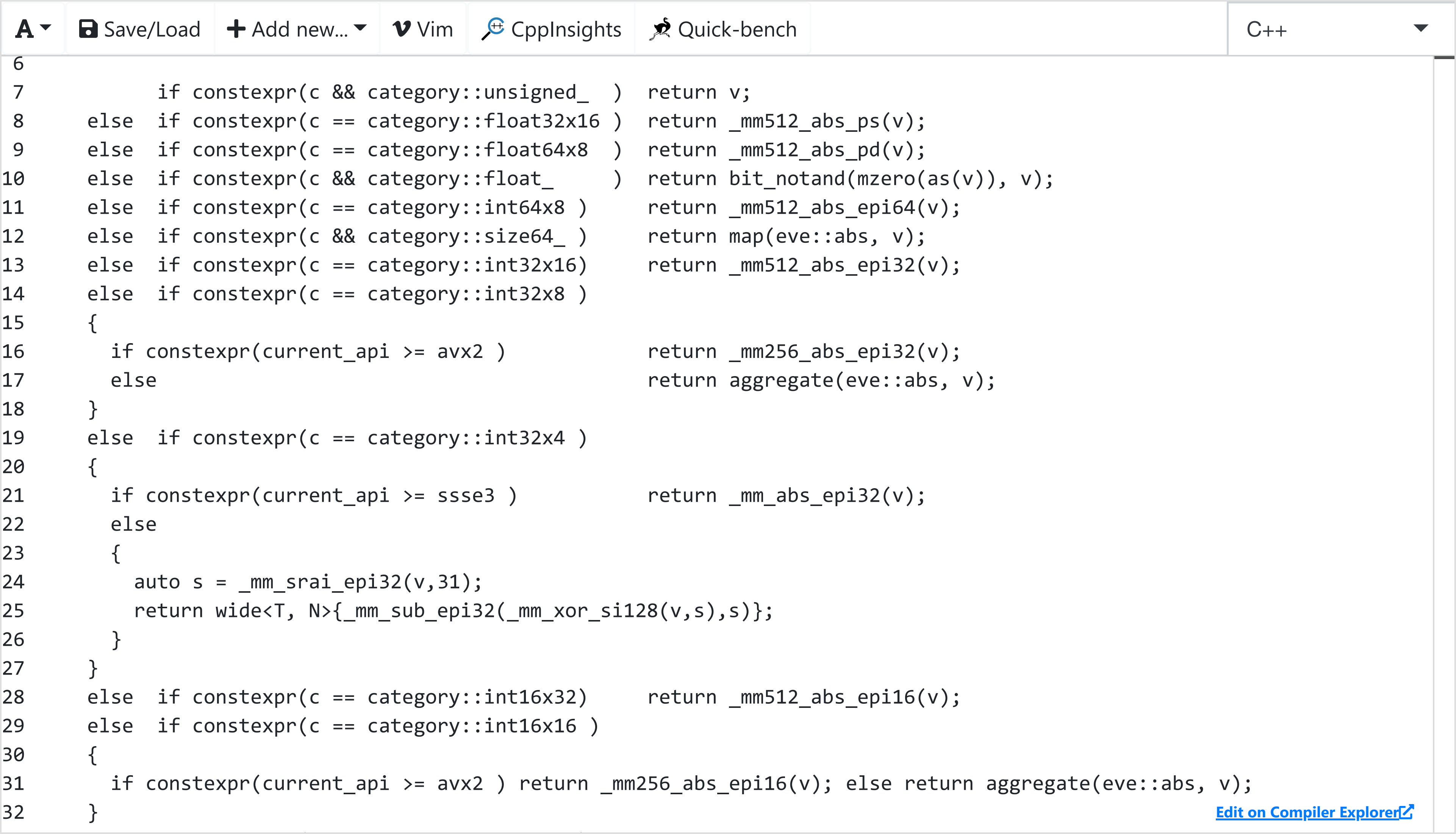
--std=c++20 -Wno-psabi -O3 -DNDEBUG

```
cos(eve::arm_abi_v0::wide<float, eve::fixed<4l> >):
        fmul    v1.4s, v0.4s, v0.4s
        adrp    x0, .LC0
        fmov    v6.4s, -5.0e-1
        ldr     q3, [x0, #:lo12:.LC0]
        adrp    x0, .LC1
        fmul    v0.4s, v1.4s, v1.4s
        ldr     q2, [x0, #:lo12:.LC1]
        adrp    x0, .LC2
        fmov    v5.4s, 1.0e+0
        ldr     q4, [x0, #:lo12:.LC2]
        adrp    x0, .LC3
```

Edit on Compiler Explorer

EVE AS A C++20 LIBRARY

EVE INTERNAL IMPLEMENTATION



```cpp
          if constexpr(c && category::unsigned_  )  return v;
    else  if constexpr(c == category::float32x16 )  return _mm512_abs_ps(v);
    else  if constexpr(c == category::float64x8  )  return _mm512_abs_pd(v);
    else  if constexpr(c && category::float_     )  return bit_notand(mzero(as(v)), v);
    else  if constexpr(c == category::int64x8 )     return _mm512_abs_epi64(v);
    else  if constexpr(c && category::size64_ )     return map(eve::abs, v);
    else  if constexpr(c == category::int32x16)     return _mm512_abs_epi32(v);
    else  if constexpr(c == category::int32x8 )
    {
      if constexpr(current_api >= avx2 )            return _mm256_abs_epi32(v);
      else                                          return aggregate(eve::abs, v);
    }
    else  if constexpr(c == category::int32x4 )
    {
      if constexpr(current_api >= ssse3 )           return _mm_abs_epi32(v);
      else
      {
        auto s = _mm_srai_epi32(v,31);
        return wide<T, N>{_mm_sub_epi32(_mm_xor_si128(v,s),s)};
      }
    }
    else  if constexpr(c == category::int16x32)     return _mm512_abs_epi16(v);
    else  if constexpr(c == category::int16x16 )
    {
      if constexpr(current_api >= avx2 ) return _mm256_abs_epi16(v); else return aggregate(eve::abs, v);
    }
```

EVE::ALGO

ALGORITHMS AVAILABLE

ALGORITHMS AVAILABLE

- all_of/any_of/none_of

ALGORITHMS AVAILABLE

- all_of/any_of/none_of
- find/find_if/find_if_not

ALGORITHMS AVAILABLE

- all_of/any_of/none_of
- find/find_if/find_if_not
- equal/mismatch

ALGORITHMS AVAILABLE

- all_of/any_of/none_of
- find/find_if/find_if_not
- equal/mismatch
- transform_inplace/transform_to

ALGORITHMS AVAILABLE

- all_of/any_of/none_of
- find/find_if/find_if_not
- equal/mismatch
- transform_inplace/transform_to
- reduce

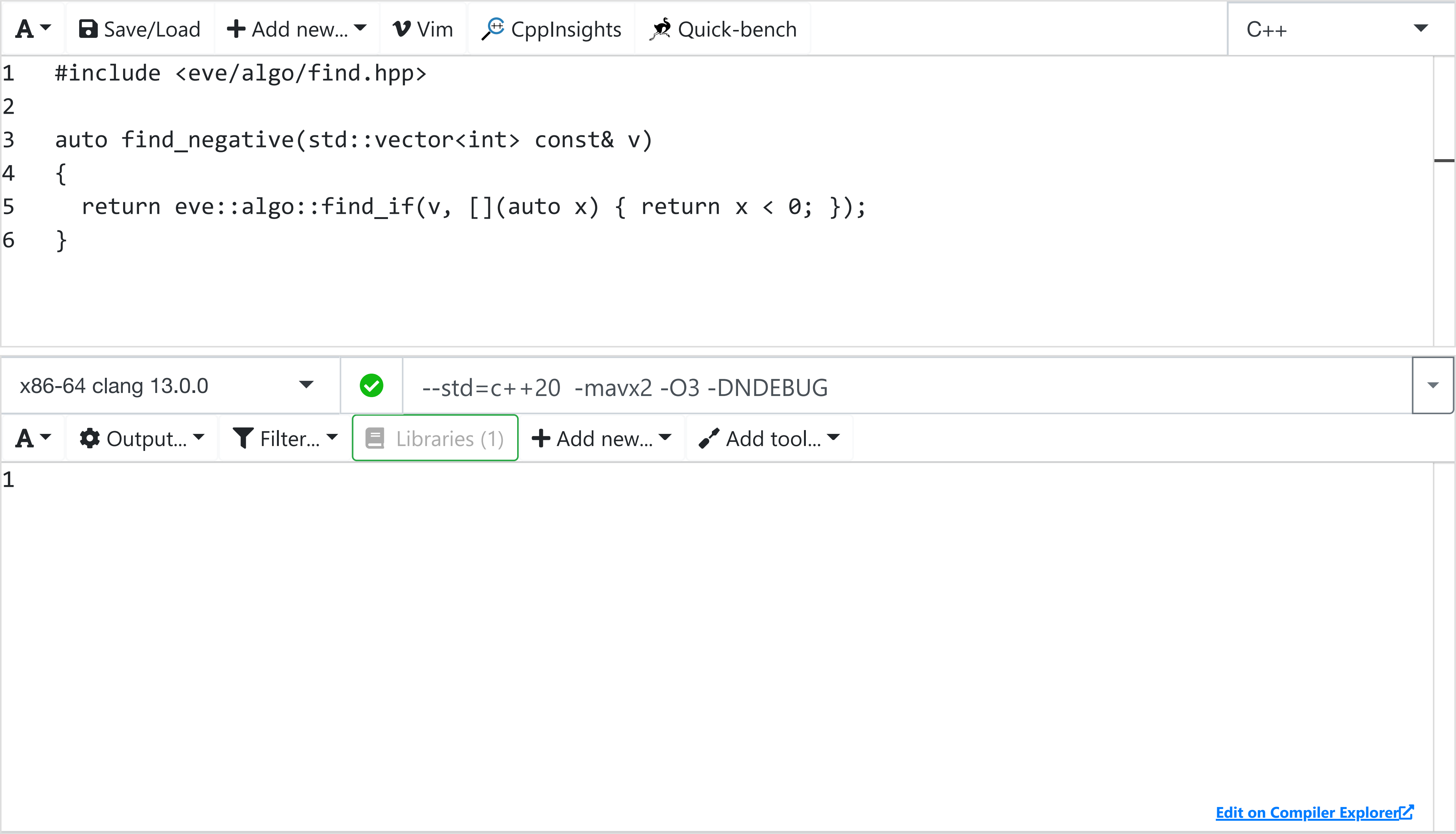
ALGORITHMS AVAILABLE

- all_of/any_of/none_of
- find/find_if/find_if_not
- equal/mismatch
- transform_inplace/transform_to
- reduce
- inclusive_scan_inplace/inclusive_scan_to

ALGORITHMS AVAILABLE

- all_of/any_of/none_of
- find/find_if/find_if_not
- equal/mismatch
- transform_inplace/transform_to
- reduce
- inclusive_scan_inplace/inclusive_scan_to
- remove/remove_if (*)

FIND A NEGATIVE NUMBER

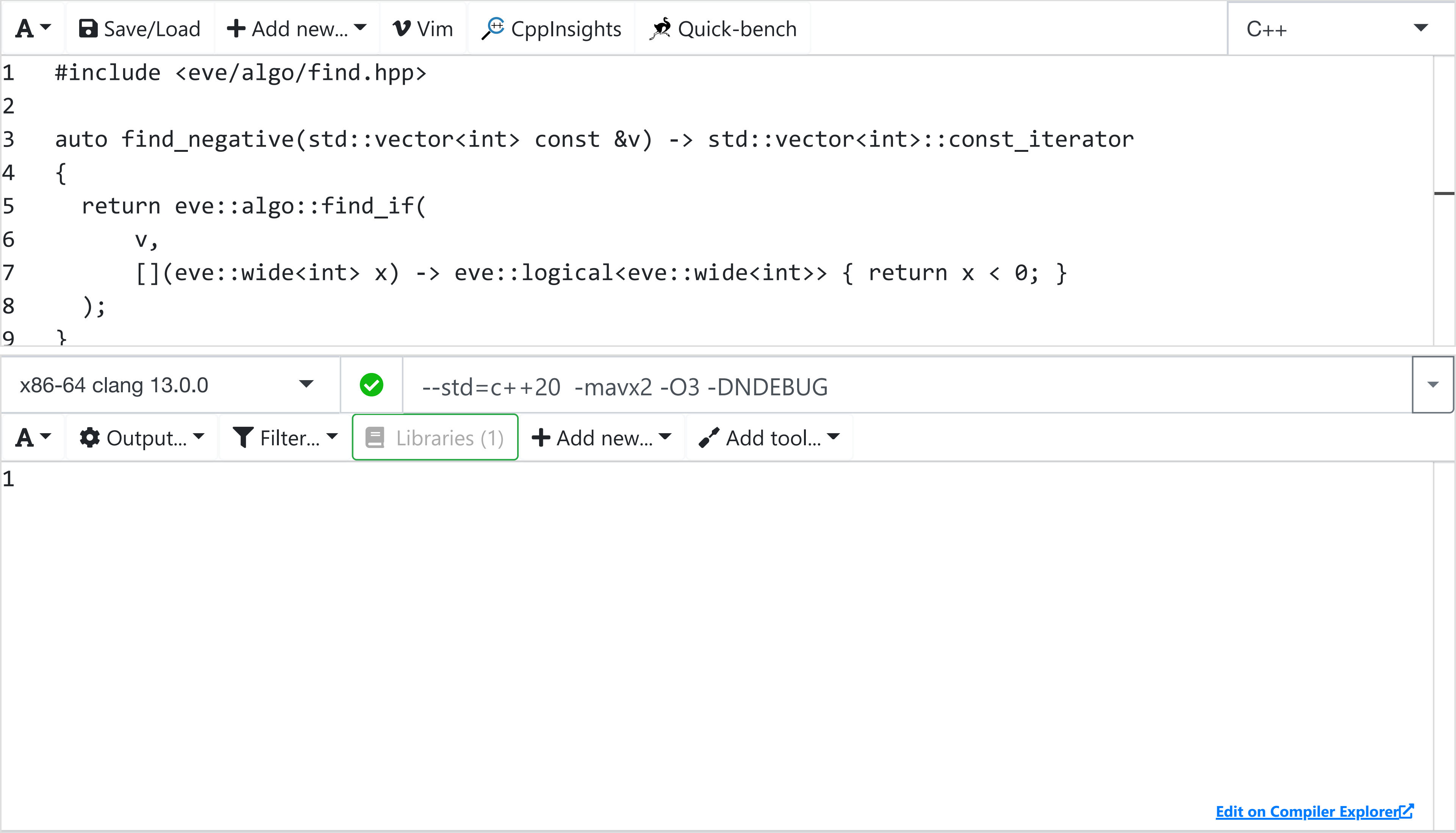


```cpp
#include <eve/algo/find.hpp>

auto find_negative(std::vector<int> const& v)
{
  return eve::algo::find_if(v, [](auto x) { return x < 0; });
}
```

x86-64 clang 13.0.0

--std=c++20 -mavx2 -O3 -DNDEBUG

FIND A NEGATIVE NUMBER

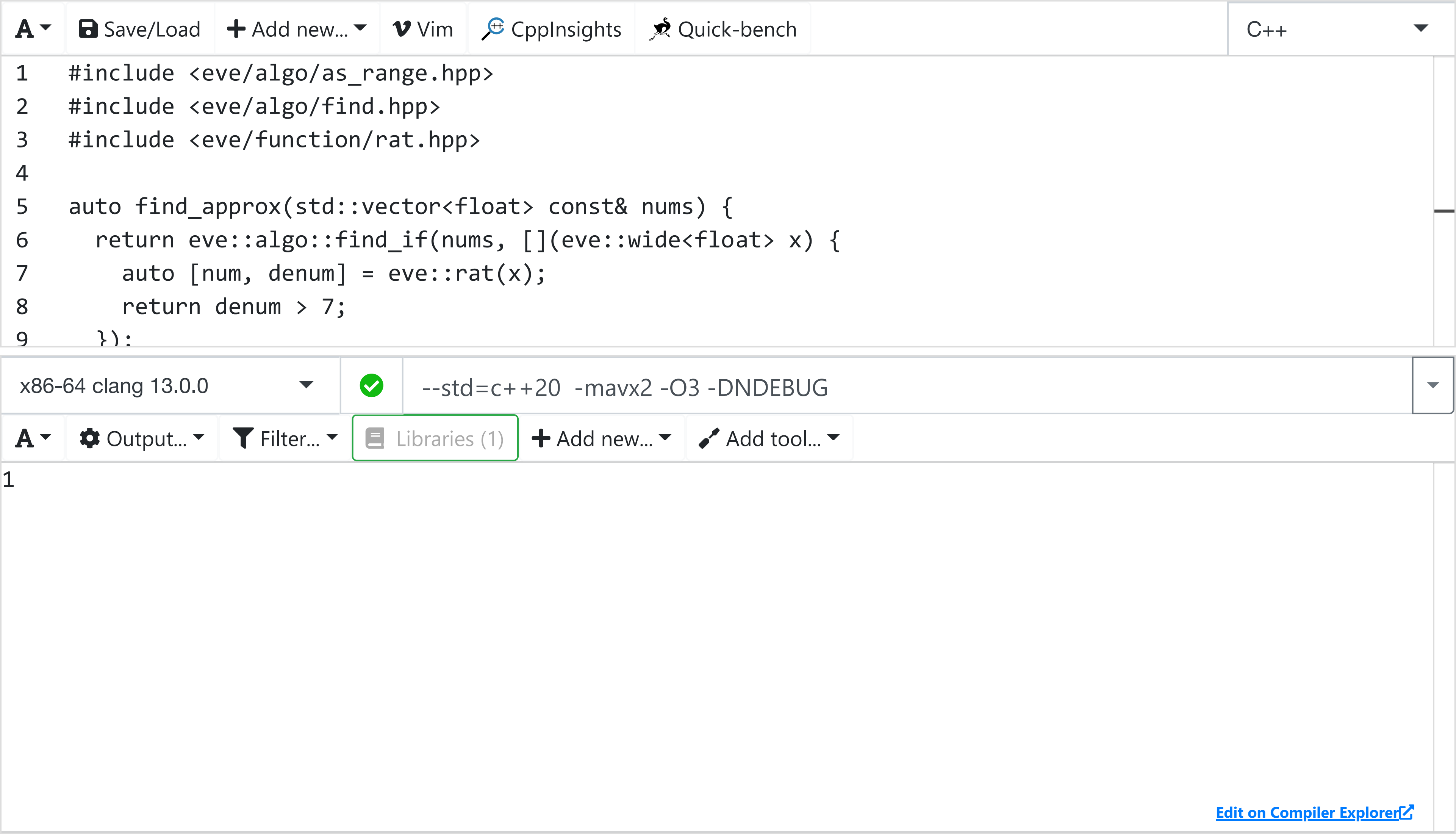


```cpp
#include <eve/algo/find.hpp>

auto find_negative(std::vector<int> const &v) -> std::vector<int>::const_iterator
{
  return eve::algo::find_if(
      v,
      [](eve::wide<int> x) -> eve::logical<eve::wide<int>> { return x < 0; }
  );
}
```

x86-64 clang 13.0.0

--std=c++20 -mavx2 -O3 -DNDEBUG

TUNING ALGORITHMS



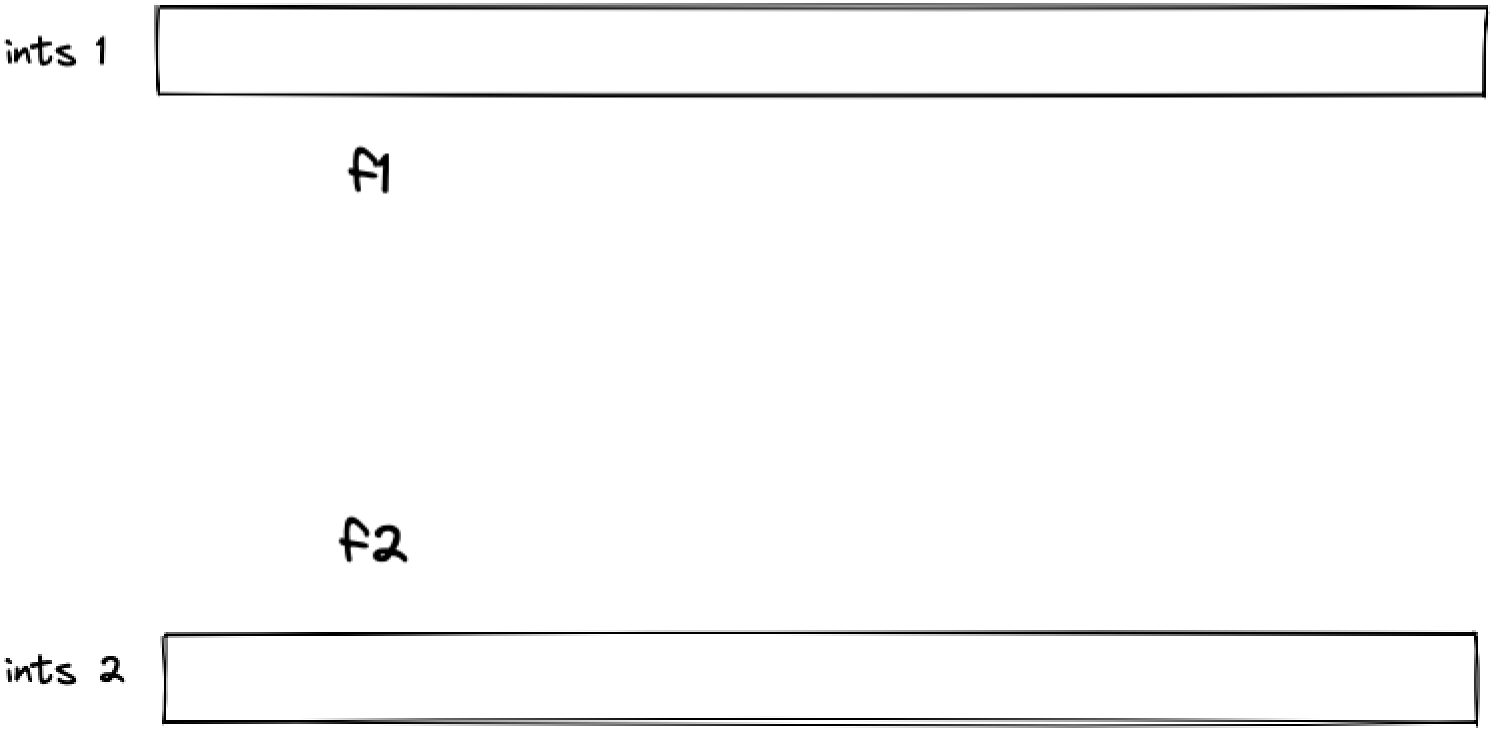
```cpp
#include <eve/algo/as_range.hpp>
#include <eve/algo/find.hpp>
#include <eve/function/rat.hpp>

auto find_approx(std::vector<float> const& nums) {
  return eve::algo::find_if(nums, [](eve::wide<float> x) {
    auto [num, denum] = eve::rat(x);
    return denum > 7;
  });
```

x86-64 clang 13.0.0

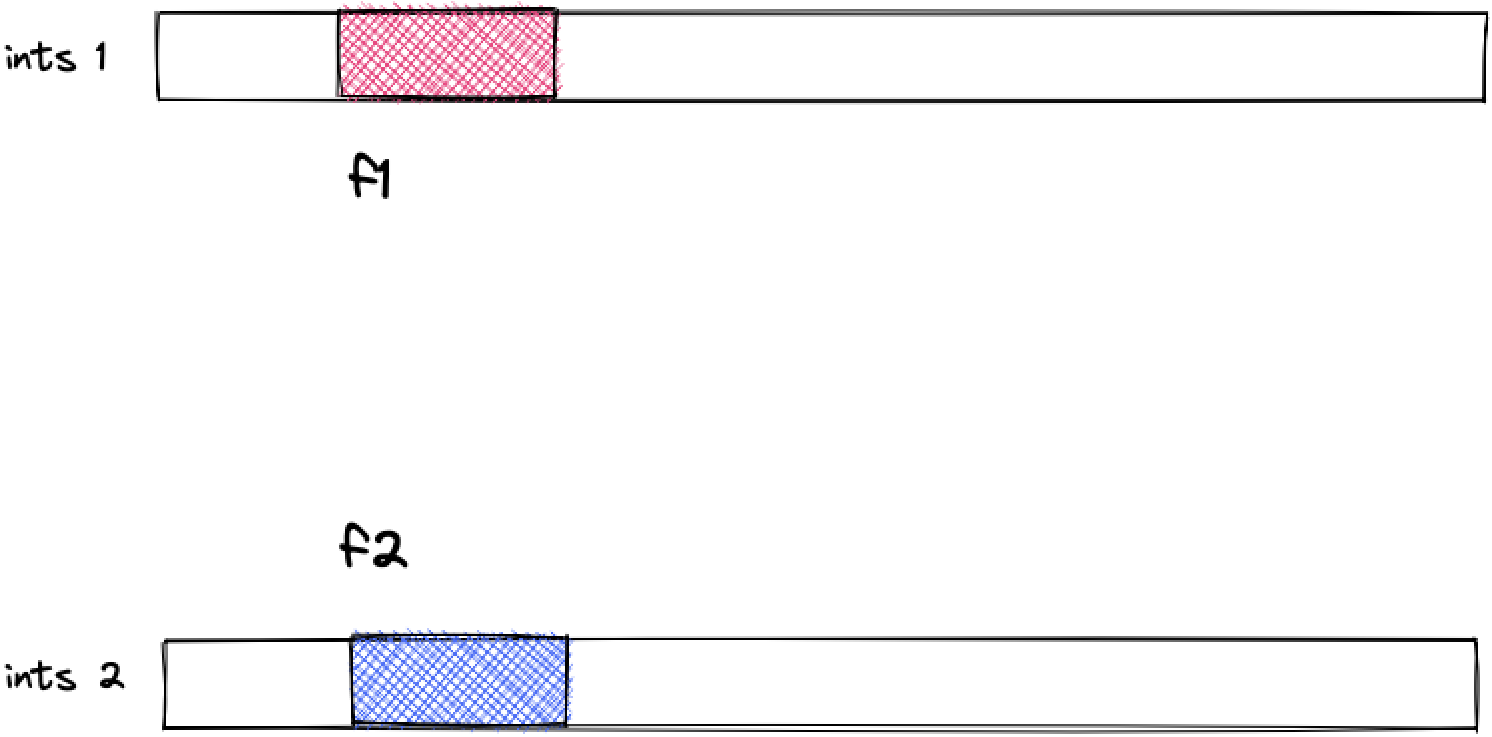
```
--std=c++20  -mavx2 -O3 -DNDEBUG
```

Mismatch is find

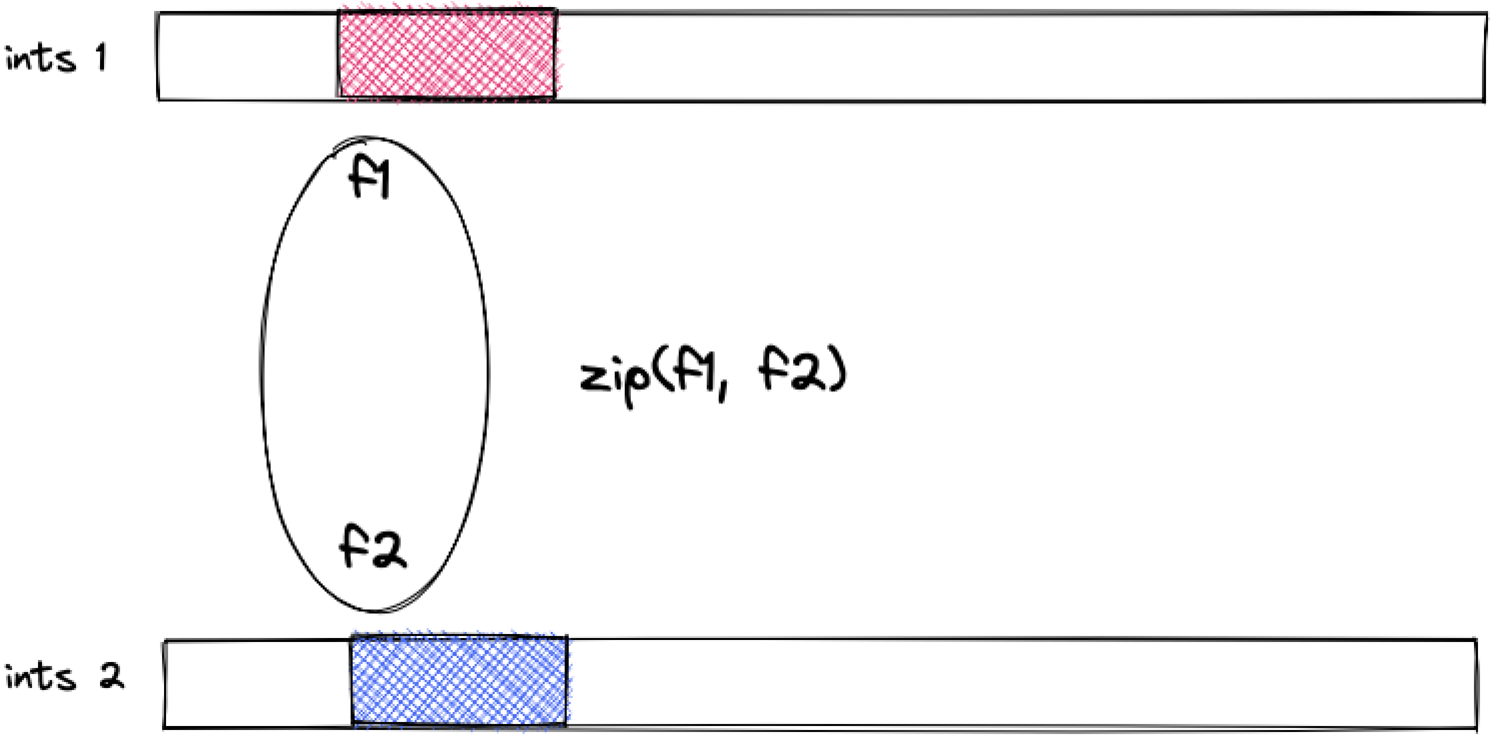
Mismatch is find



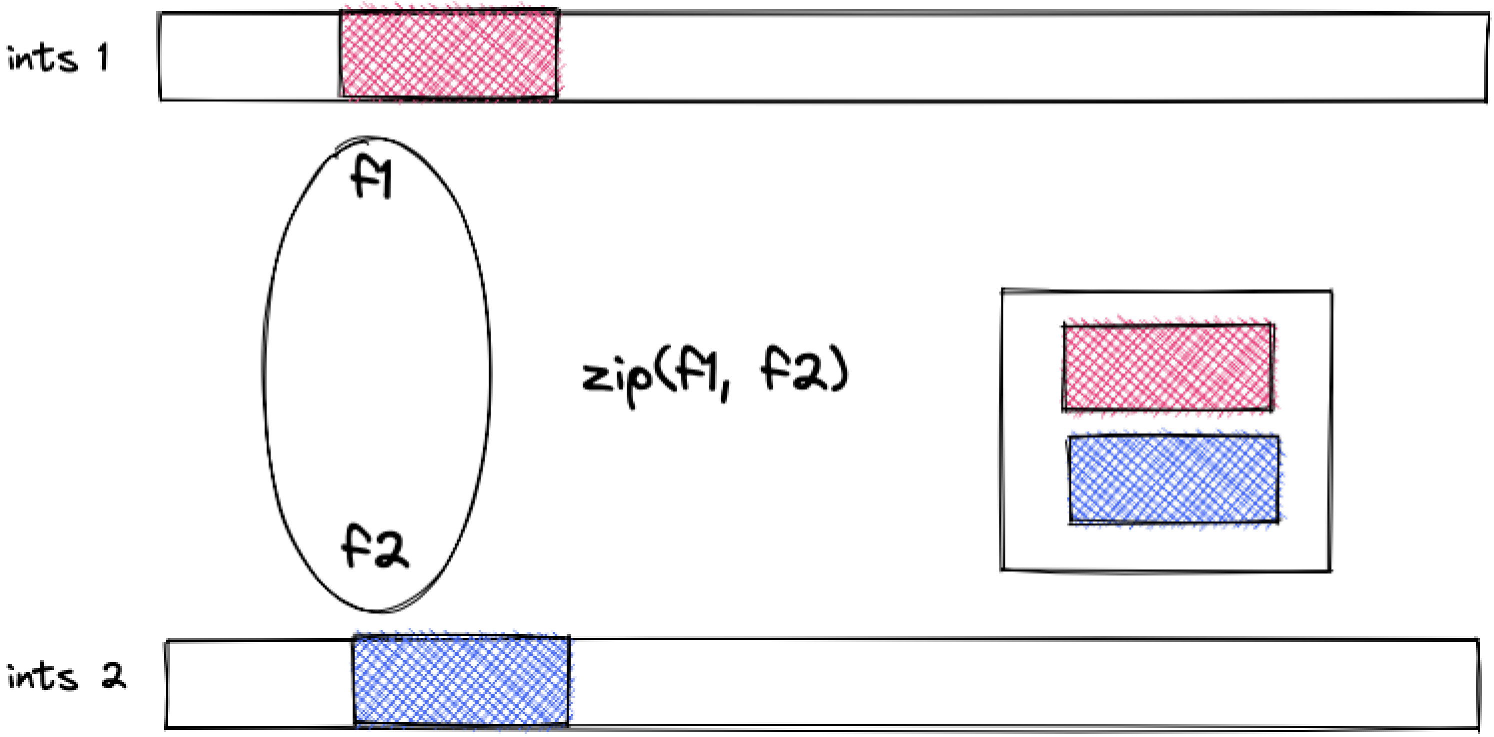
Mismatch is find



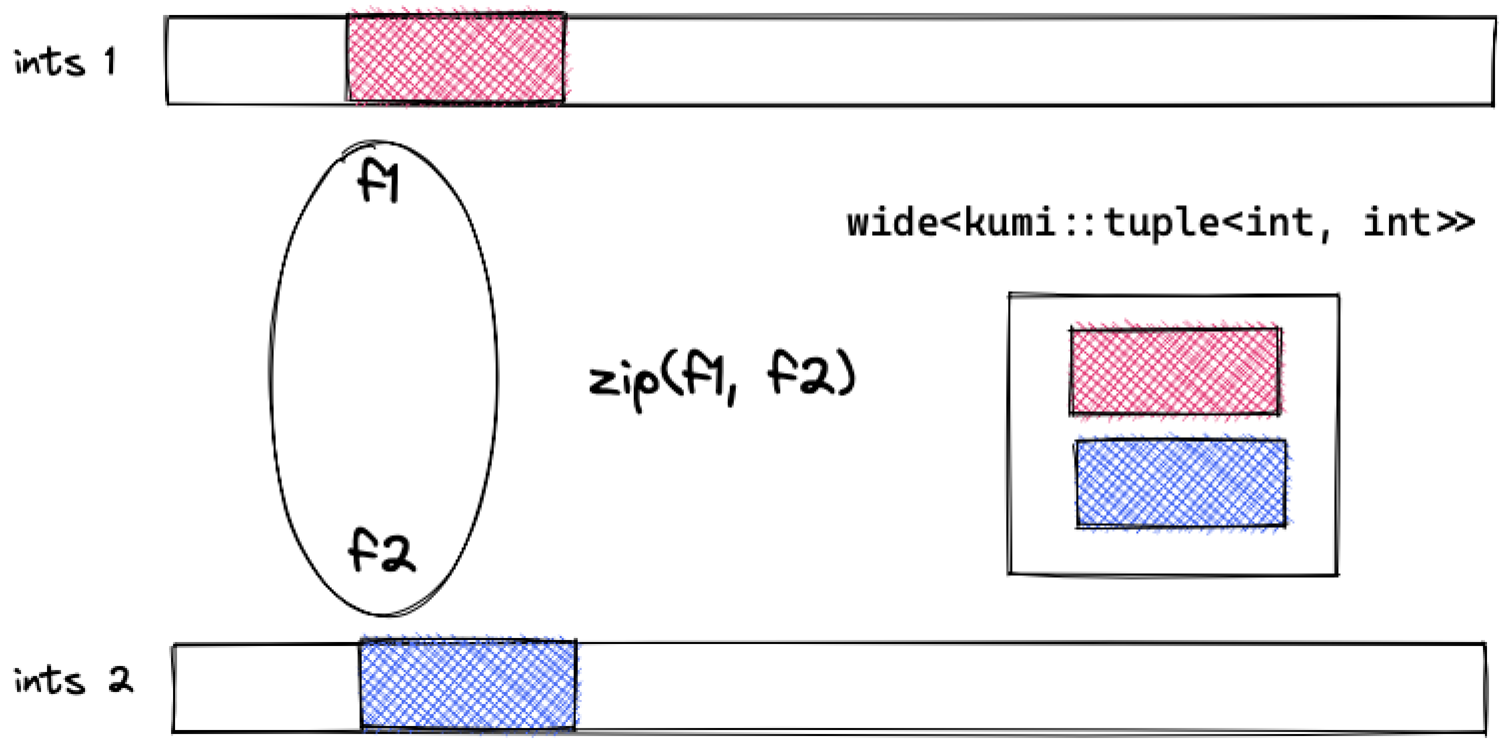
Mismatch is find



Mismatch is find



Mismatch is find



ints 1

f1

f2

ints 2

zip(f1, f2)

`wide<kumi::tuple<int, int>>`

MISMATCH INTERFACE

```
auto std::mismatch(I1 f1, I1 l1, I2 f2, P p) -> std::pair

auto mismatch(R1&& r1, R2&& r2, P p) -> zip_iterator
  requires zip_to_range<R1, R2>

auto mismatch(zipped_range_pair auto&& r, P p) -> zip_iterator
```

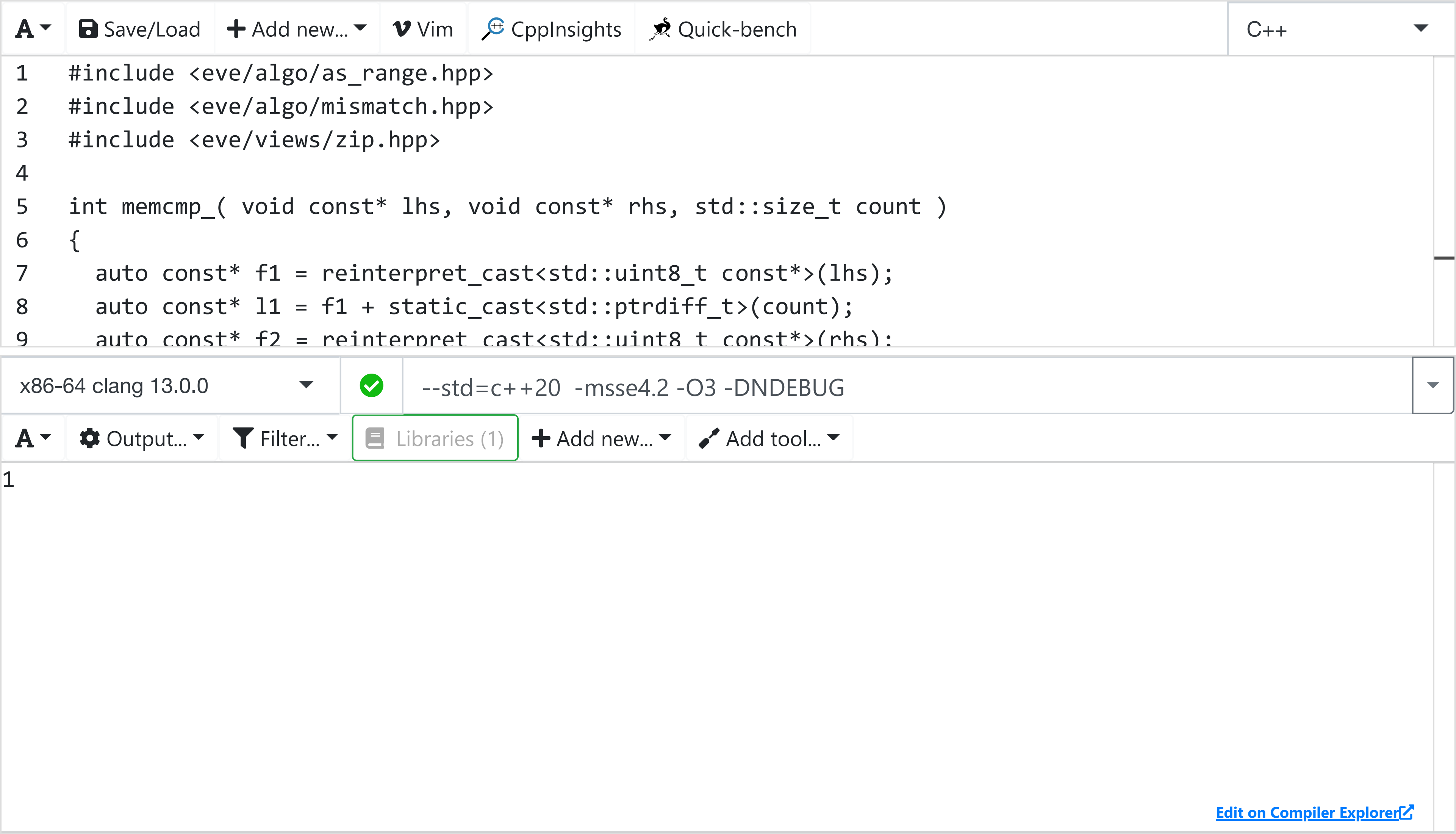
MISMATCH INTERFACE

```
auto std::mismatch(I1 f1, I1 l1, I2 f2, P p) -> std::pair

auto mismatch(R1&& r1, R2&& r2, P p) -> zip_iterator
  requires zip_to_range<R1, R2>

auto mismatch(zipped_range_pair auto&& r, P p) -> zip_iterator
```

MISMATCH INTERFACE

```
auto std::mismatch(I1 f1, I1 l1, I2 f2, P p) -> std::pair

auto mismatch(R1&& r1, R2&& r2, P p) -> zip_iterator
  requires zip_to_range<R1, R2>

auto mismatch(zipped_range_pair auto&& r, P p) -> zip_iterator
```

MEMCMP



```cpp
#include <eve/algo/as_range.hpp>
#include <eve/algo/mismatch.hpp>
#include <eve/views/zip.hpp>

int memcmp_( void const* lhs, void const* rhs, std::size_t count )
{
  auto const* f1 = reinterpret_cast<std::uint8_t const*>(lhs);
  auto const* l1 = f1 + static_cast<std::ptrdiff_t>(count);
  auto const* f2 = reinterpret_cast<std::uint8_t const*>(rhs);
```

```
--std=c++20  -msse4.2 -O3 -DNDEBUG
```

```
1
```

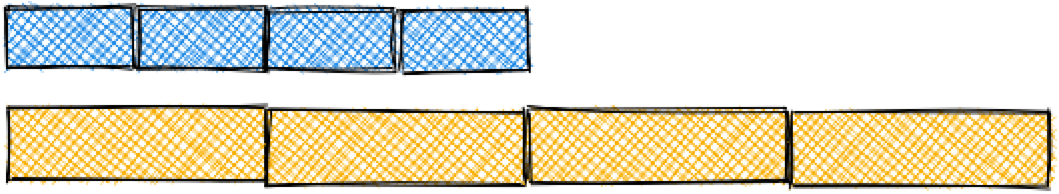
Zip different types

Zip different types

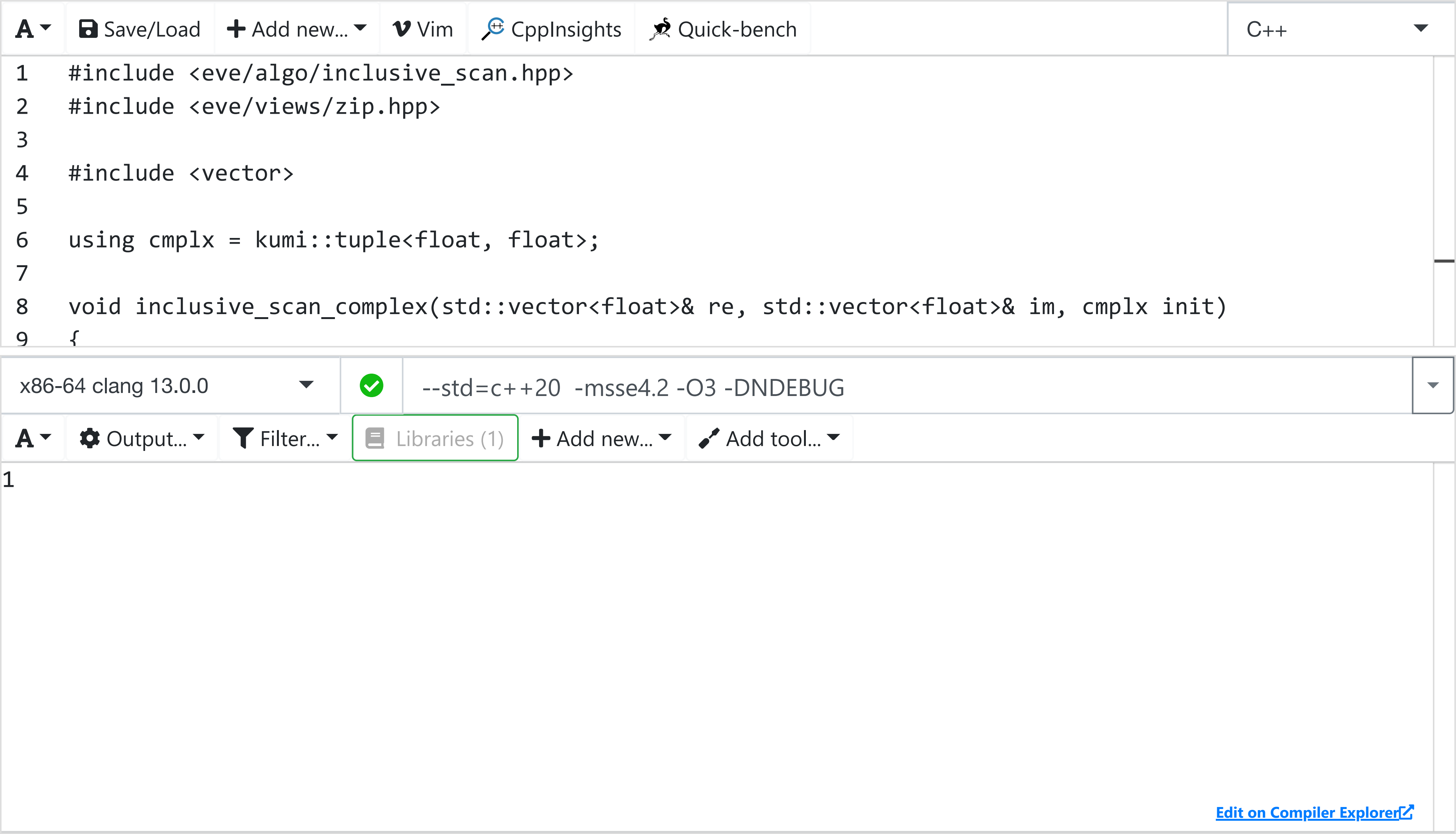
```
wide<kumi::tuple<int, double>>
```

Zip different types

```
wide<kumi::tuple<int, double>>
```



INCLUSIVE_SCAN COMPLEX NUMBERS



```cpp
#include <eve/algo/inclusive_scan.hpp>
#include <eve/views/zip.hpp>

#include <vector>

using cmplx = kumi::tuple<float, float>;

void inclusive_scan_complex(std::vector<float>& re, std::vector<float>& im, cmplx init)
{
```

```
--std=c++20  -msse4.2 -O3 -DNDEBUG
```

COLLECT INDEXES EXAMPLE

COLLECT INDEXES EXAMPLE

- Max De Marzi's blog on RageDB

COLLECT INDEXES EXAMPLE

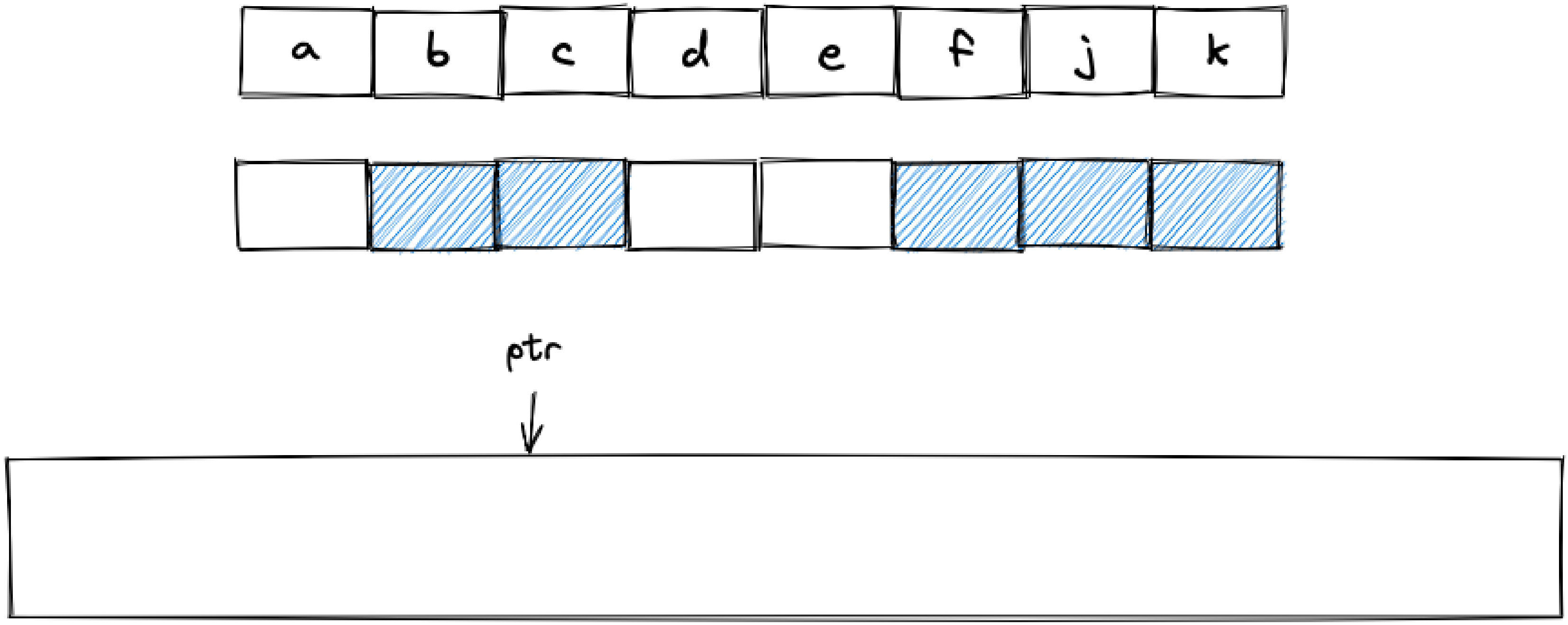
- Max De Marzi's blog on RageDB
- Collect indexes of elements matching the predicate

COLLECT INDEXES EXAMPLE

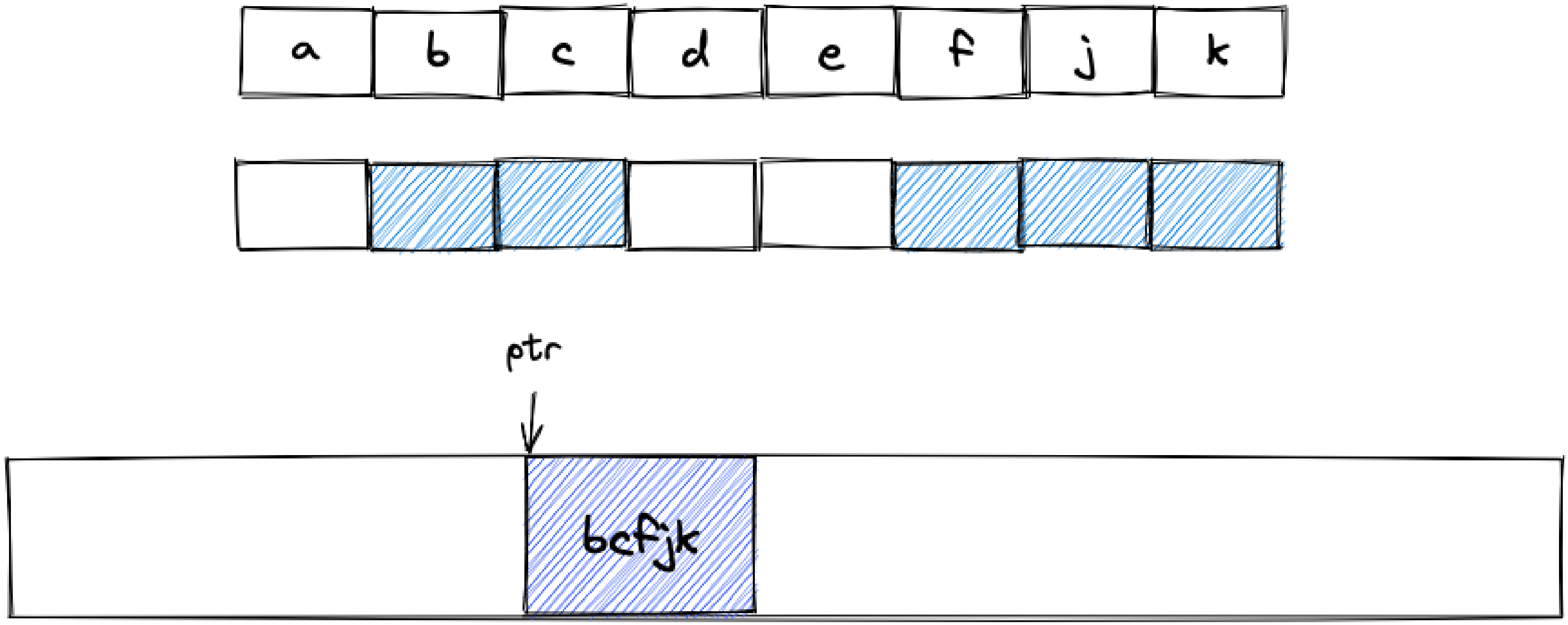
- Max De Marzi's blog on RageDB
- Collect indexes of elements matching the predicate
- requests per second went up 5.75 times

```
unsafe(compress_store)(wide, logical, T*) -> T*
```

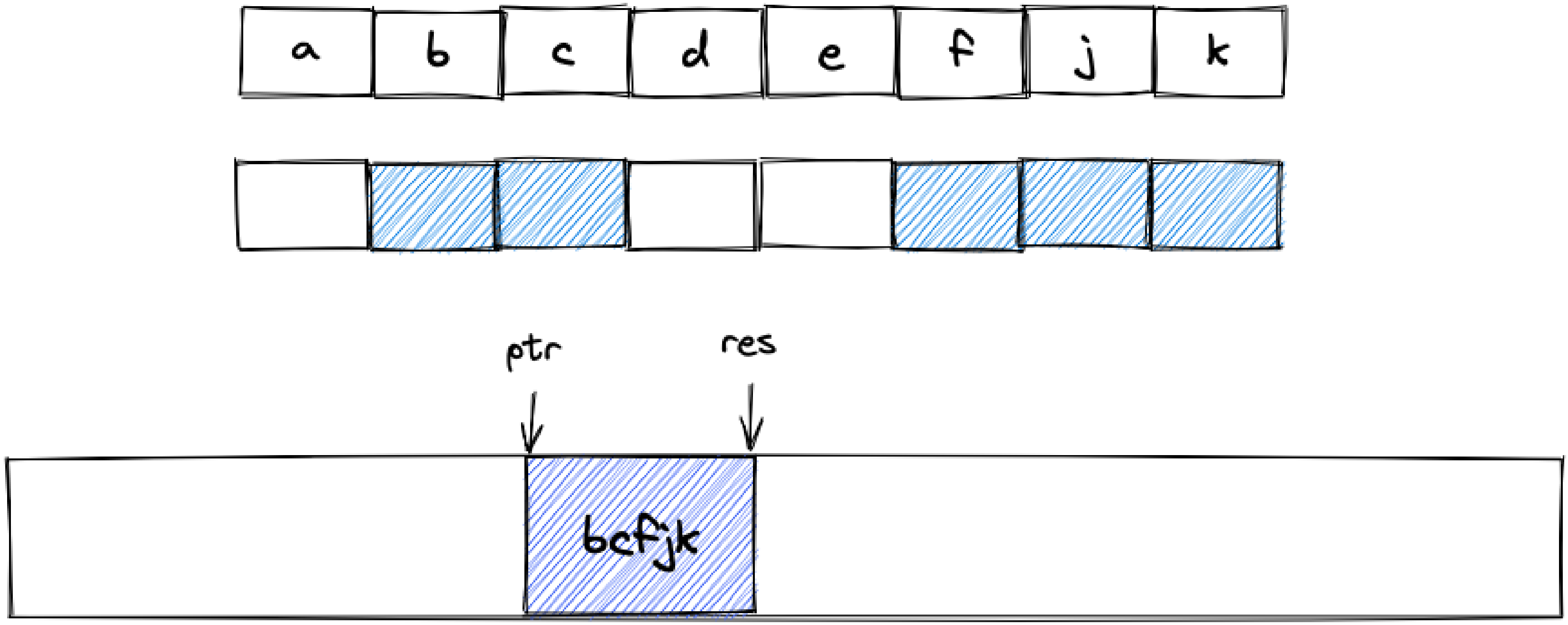
unsafe(compress_store)(wide, logical, T*) -> T*



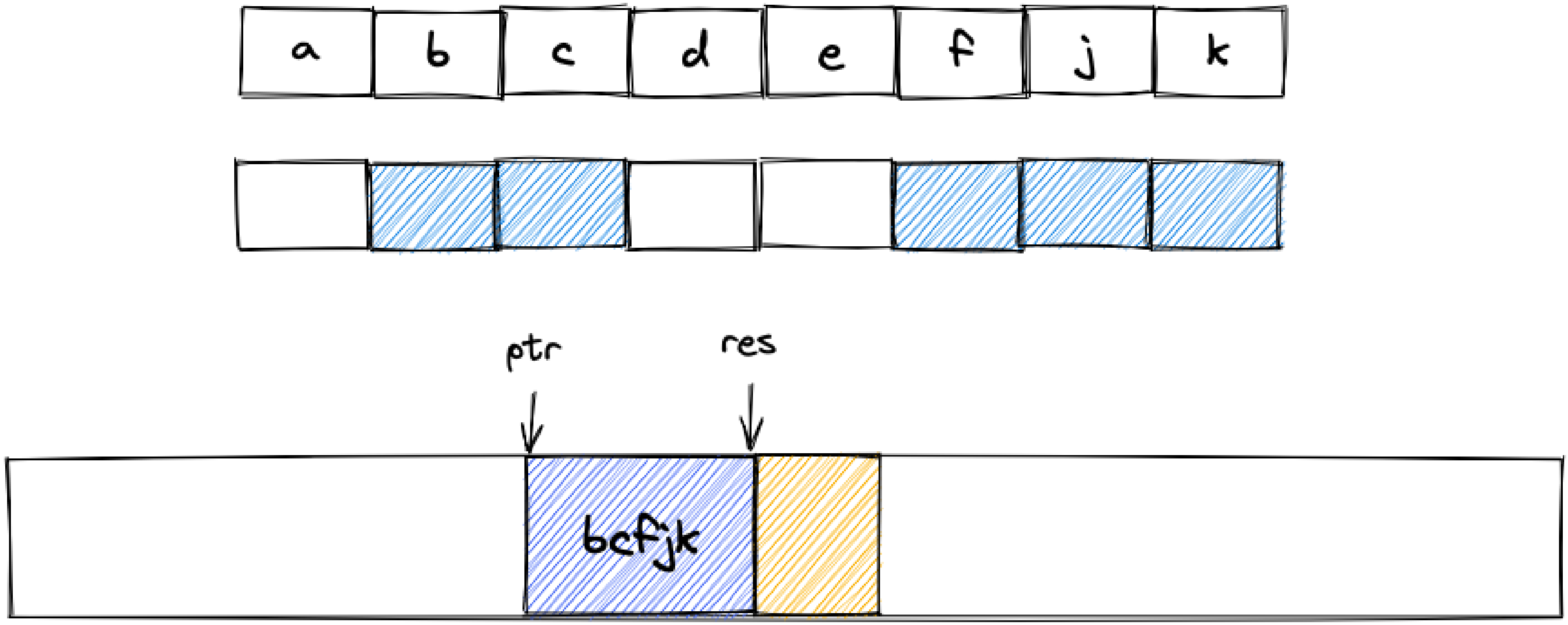
unsafe(compress_store)(wide, logical, T*) -> T*



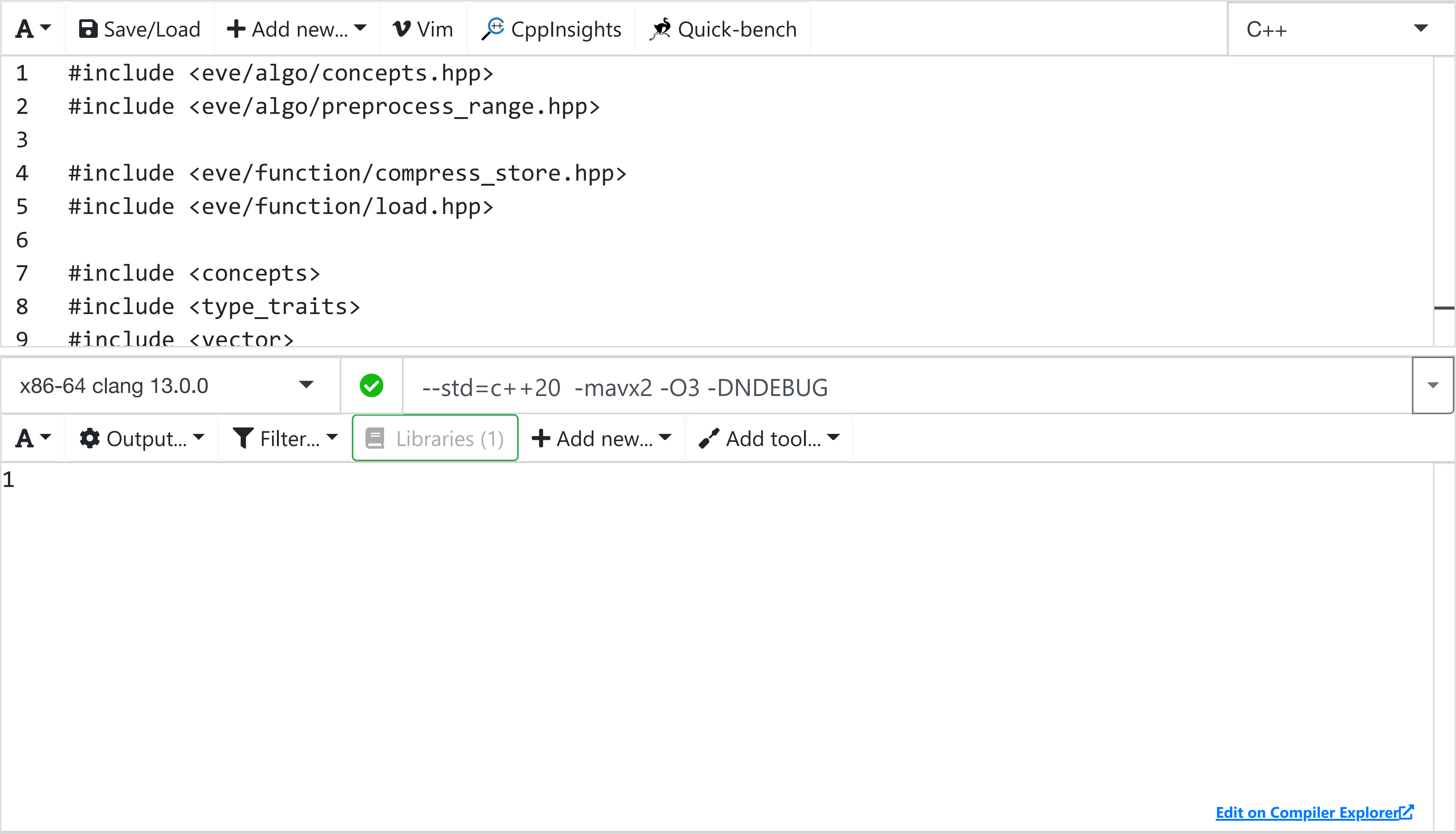
unsafe(compress_store)(wide, logical, T*) -> T*



unsafe(compress_store)(wide, logical, T*) -> T*



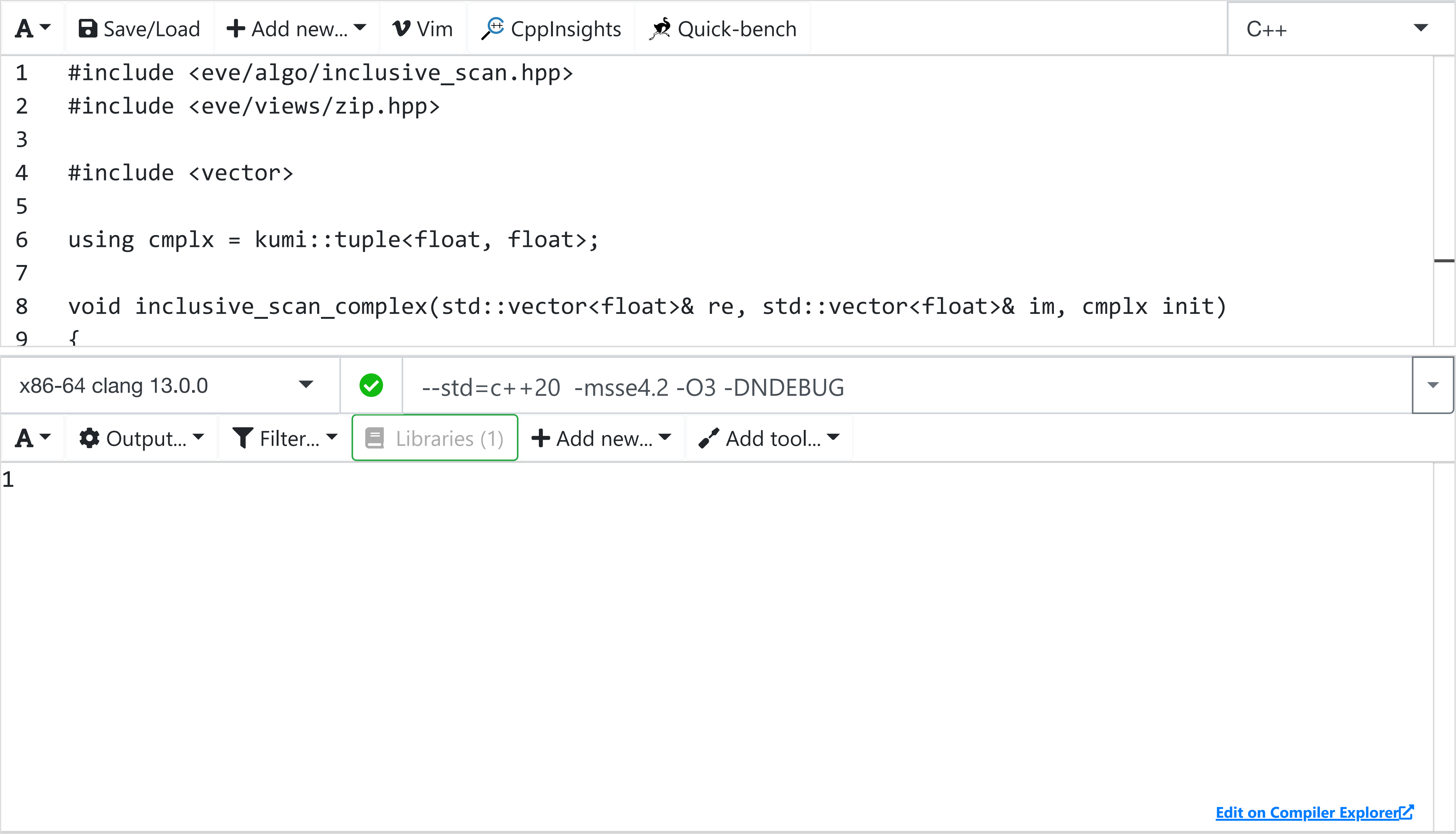
COLLECT INDEXES



```
#include <eve/algo/concepts.hpp>
#include <eve/algo/preprocess_range.hpp>

#include <eve/function/compress_store.hpp>
#include <eve/function/load.hpp>

#include <concepts>
#include <type_traits>
#include <vector>
```

```
--std=c++20  -mavx2 -O3 -DNDEBUG
```

BACK TO INCLUSIVE_SCAN COMPLEX NUMBERS



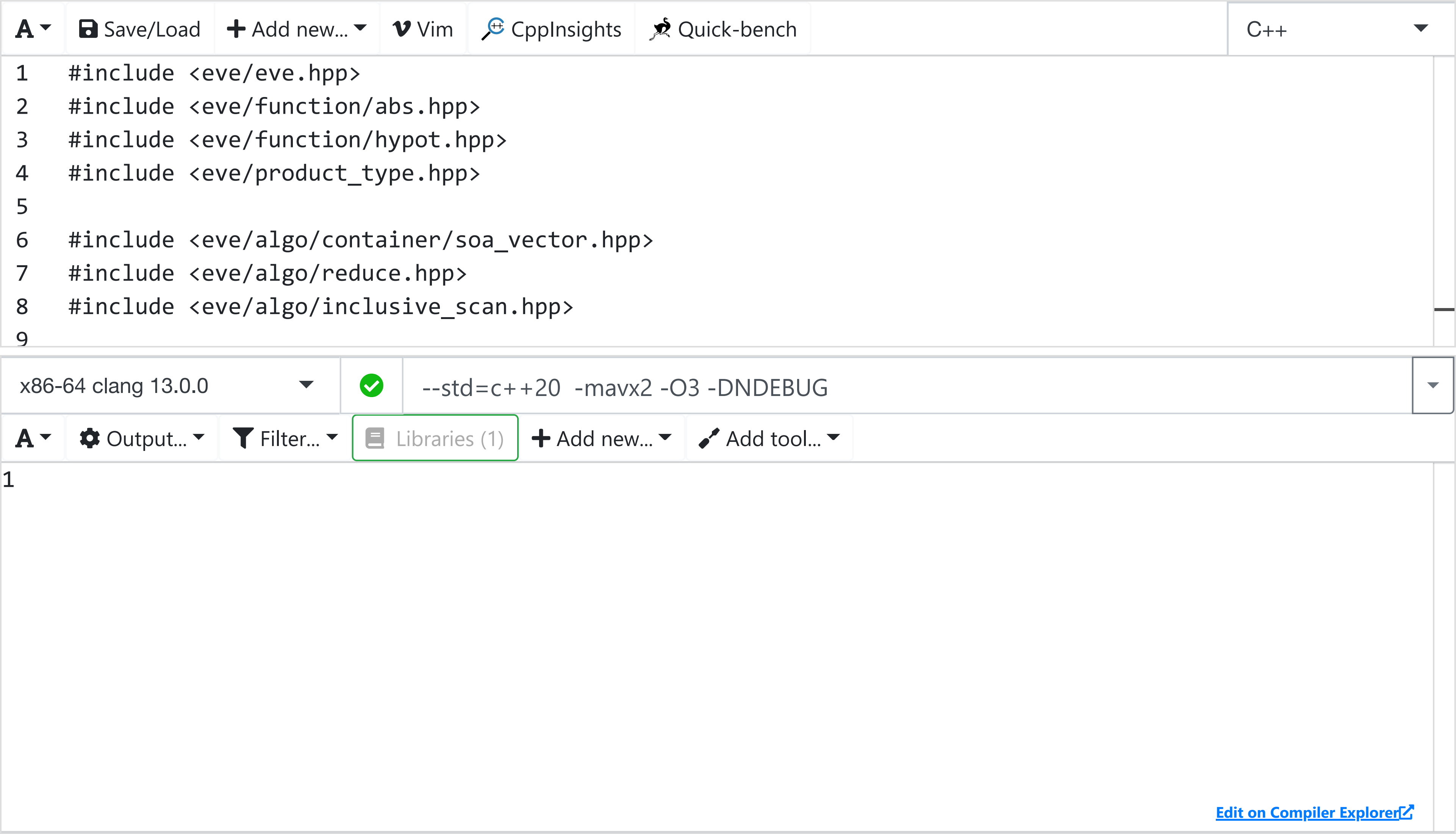
```cpp
#include <eve/algo/inclusive_scan.hpp>
#include <eve/views/zip.hpp>

#include <vector>

using cmplx = kumi::tuple<float, float>;

void inclusive_scan_complex(std::vector<float>& re, std::vector<float>& im, cmplx init)
{
```

x86-64 clang 13.0.0

--std=c++20 -msse4.2 -O3 -DNDEBUG

```
1
```

OBJECTIVELY BETTER

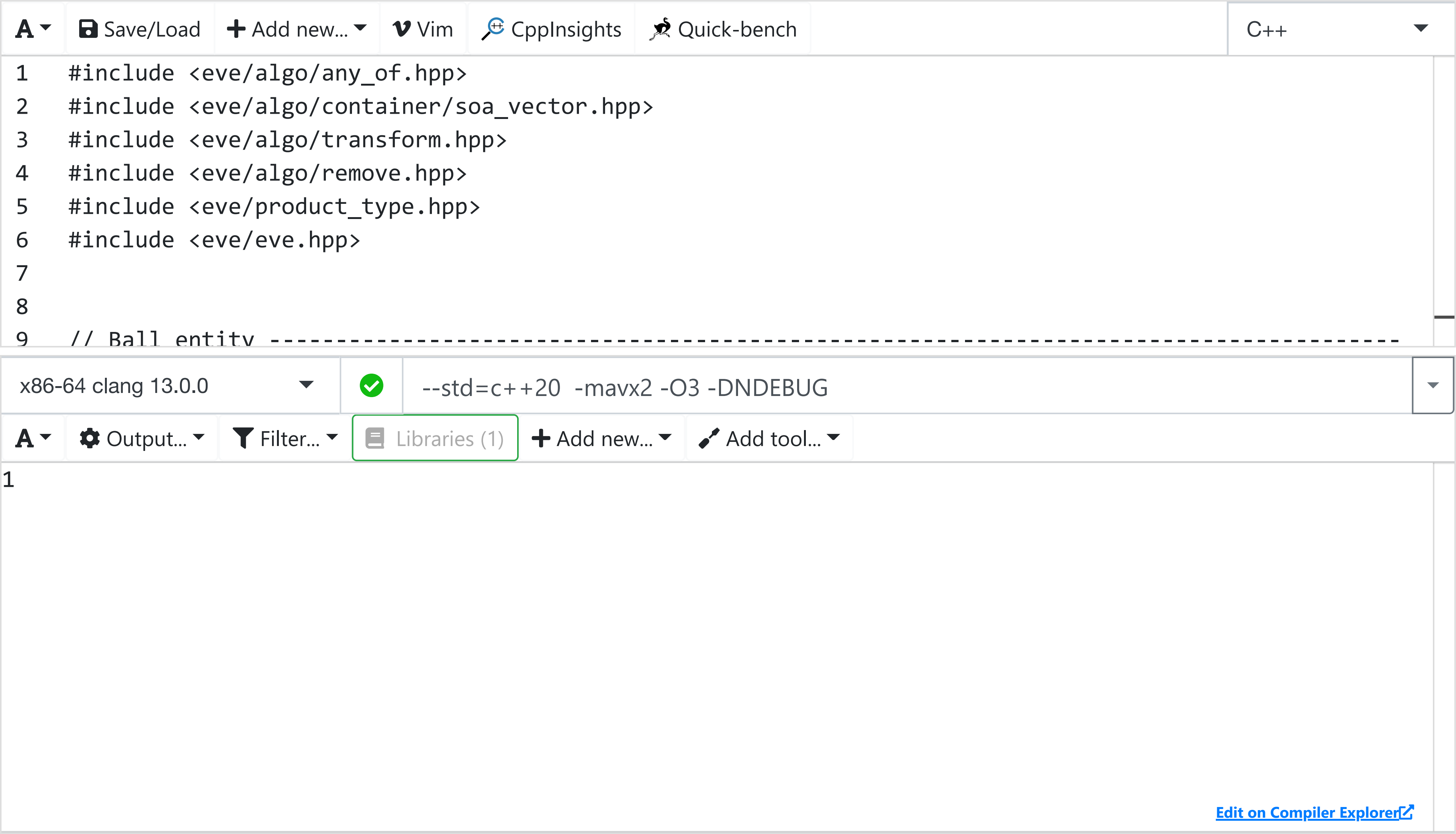


```
#include <eve/eve.hpp>
#include <eve/function/abs.hpp>
#include <eve/function/hypot.hpp>
#include <eve/product_type.hpp>

#include <eve/algo/container/soa_vector.hpp>
#include <eve/algo/reduce.hpp>
#include <eve/algo/inclusive_scan.hpp>
```

x86-64 clang 13.0.0

--std=c++20 -mavx2 -O3 -DNDEBUG

DATA-ORIENTED DESIGN



```cpp
#include <eve/algo/any_of.hpp>
#include <eve/algo/container/soa_vector.hpp>
#include <eve/algo/transform.hpp>
#include <eve/algo/remove.hpp>
#include <eve/product_type.hpp>
#include <eve/eve.hpp>


// Ball entity ------------------------------------------------------------------------
```

x86-64 clang 13.0.0

--std=c++20 -mavx2 -O3 -DNDEBUG

```
1
```

Edit on Compiler Explorer

PRACTICALITIES

WHAT ABOUT BUGS?

WHAT ABOUT BUGS?

YES

API STABILITY?

API STABILITY?

NO

RECOMMENDED WORKFLOW

RECOMMENDED WORKFLOW

- focus on the critical components

RECOMMENDED WORKFLOW

- focus on the critical components
- build a standalone library

RECOMMENDED WORKFLOW

- focus on the critical components
- build a standalone library
- dynamic linking

RECOMMENDED WORKFLOW

- focus on the critical components
- build a standalone library
- dynamic linking
- contact us

SIMILAR LIBRARIES

SIMILAR LIBRARIES

- intrinsics wrappers Vc(std::simd), xsimd, tsimd

SIMILAR LIBRARIES

- intrinsics wrappers Vc(std::simd), xsimd, tsimd
- SIMD everywhere

MENTIONS

MENTIONS

- Jean-Thierry Lapresté

MENTIONS

- Jean-Thierry Lapresté
- Stack Overflow: @aqrit, @Peter Cordes, @Z boson, @Stephen Canon

MENTIONS

- Jean-Thierry Lapresté
- Stack Overflow: @aqrit, @Peter Cordes, @Z boson, @Stephen Canon
- Unity for soa_vector idea

OTHER TALKS

OTHER TALKS

- My First SIMD (Meeting C++ 2020)

CONTACT INFORMATION

- slides: tinyurl.com/eve-simd-2021
- github.com/jfalcou/eve (discussions/issues)
- Discord
- cpplang slack: jfalcou, dyaroshev
- email: joel.falcou@lri.fr, denis.yaroshevskij@gmail.com
- twitter: @CppSpelunker, @dyaroshev