MOVING AN INT IS SLOW

DEBUG PEFORMANCE MATTERS!

Vittorio Romeo

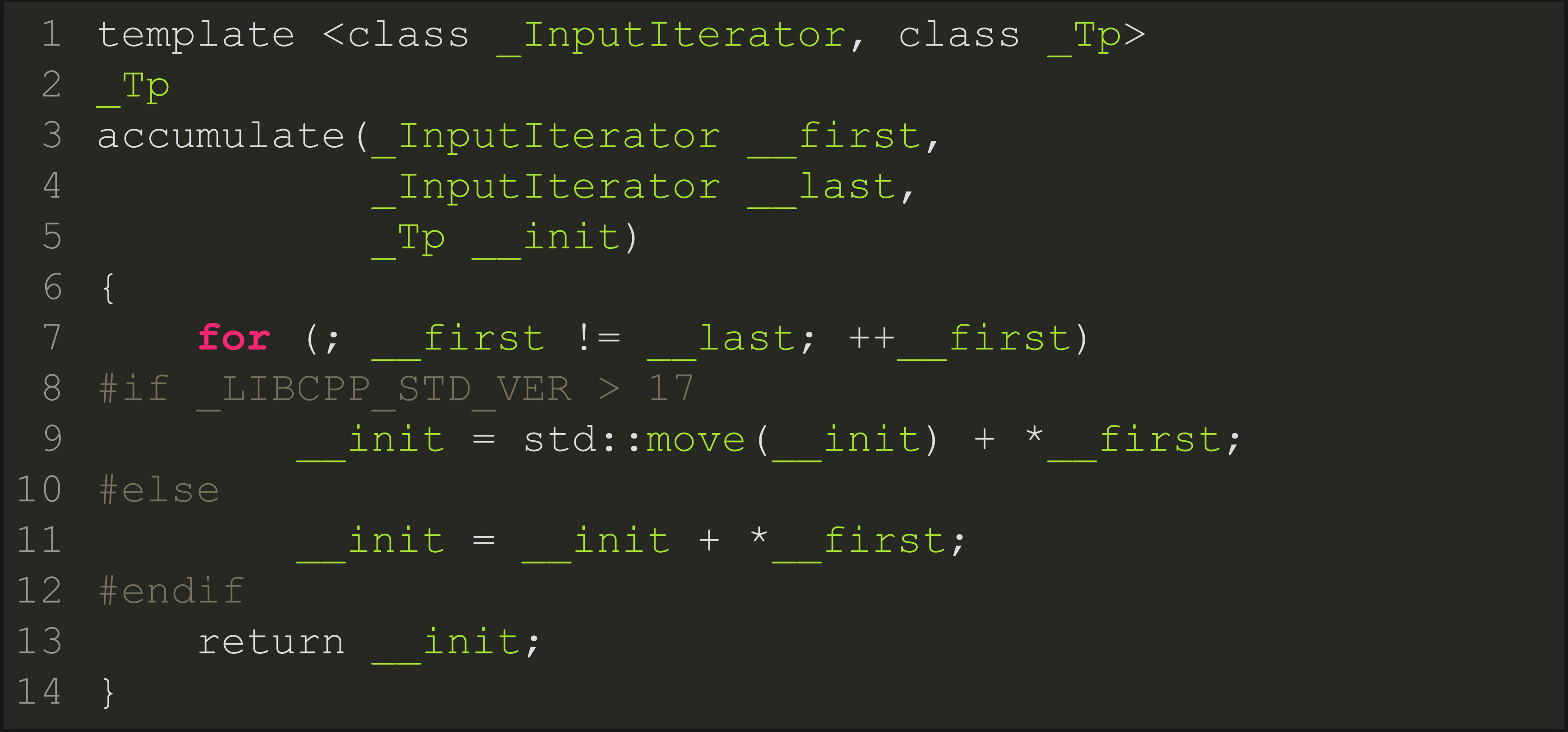
mail@vittorioromeo.com

@supahvee1234

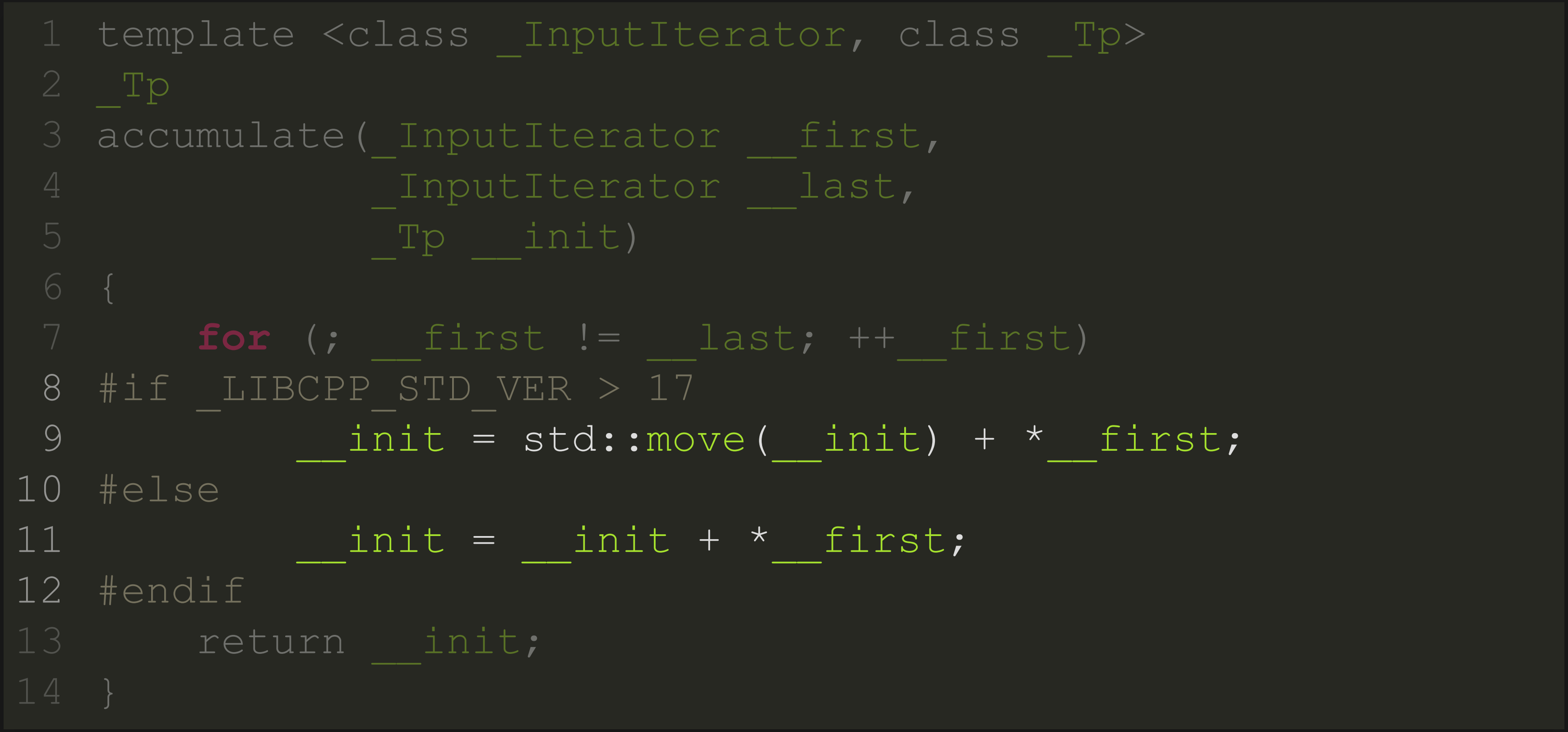
https://github.com/vittorioromeo/accu2022

```cpp
int accumulate_range(int* begin, int* end)
{
    return std::accumulate(begin, end, 0);
}
```

- GCC and Clang, compiling with `-O0`
  - `accumulate_range` runs 1.4x slower in C++20 compared to C++17
- Why?



```
template <class _InputIterator, class _Tp>
_Tp
accumulate(_InputIterator __first,
           _InputIterator __last,
           _Tp __init)
{
    for (; __first != __last; ++__first)
#if _LIBCPP_STD_VER > 17
        __init = std::move(__init) + *__first;
#else
        __init = __init + *__first;
#endif
    return __init;
}
```



```cpp
template <class _InputIterator, class _Tp>
_Tp
accumulate(_InputIterator __first,
           _InputIterator __last,
           _Tp __init)
{
    for (; __first != __last; ++__first)
#if _LIBCPP_STD_VER > 17
        __init = std::move(__init) + *__first;
#else
        __init = __init + *__first;
#endif
    return __init;
}
```

- Wait, is `std::move` adding run-time overhead?
- Isn’t `std::move` just a cast?

```
template <class _Tp>
[[nodiscard]] inline constexpr
std::remove_reference_t<_Tp>&& move(_Tp&& __t) noexcept
{
    return static_cast<std::remove_reference_t<_Tp>&&>(__t);
}
```

- Semantically, it is just a cast
- To the compiler, it is just another *function call*
- I.e. overhead unless inlining happens
  - It doesn’t in `-O0`

The same issue applies to many functions:

- `std::move`
- `std::forward`
- `std::as_const`
- `std::as_underlying`
- `std::vector<T>::iterator::operator*`
- `std::vector<T>::iterator::operator++`
- `std::unique_ptr<T>::operator*`
- *etc…*

In some fields, debug performance is very important:

- E.g. games can’t be unplayable in debug mode

People are discouraged from using Modern C++

- So they write C-like code…
- …which has more bugs…
- …which needs more debugging.

WHAT CAN WE DO?

- `-Og` doesn’t cut it
    - Sometimes optimizes too much
    - For Clang, it’s the same as `-O1`
    - MSVC doesn’t have an equivalent

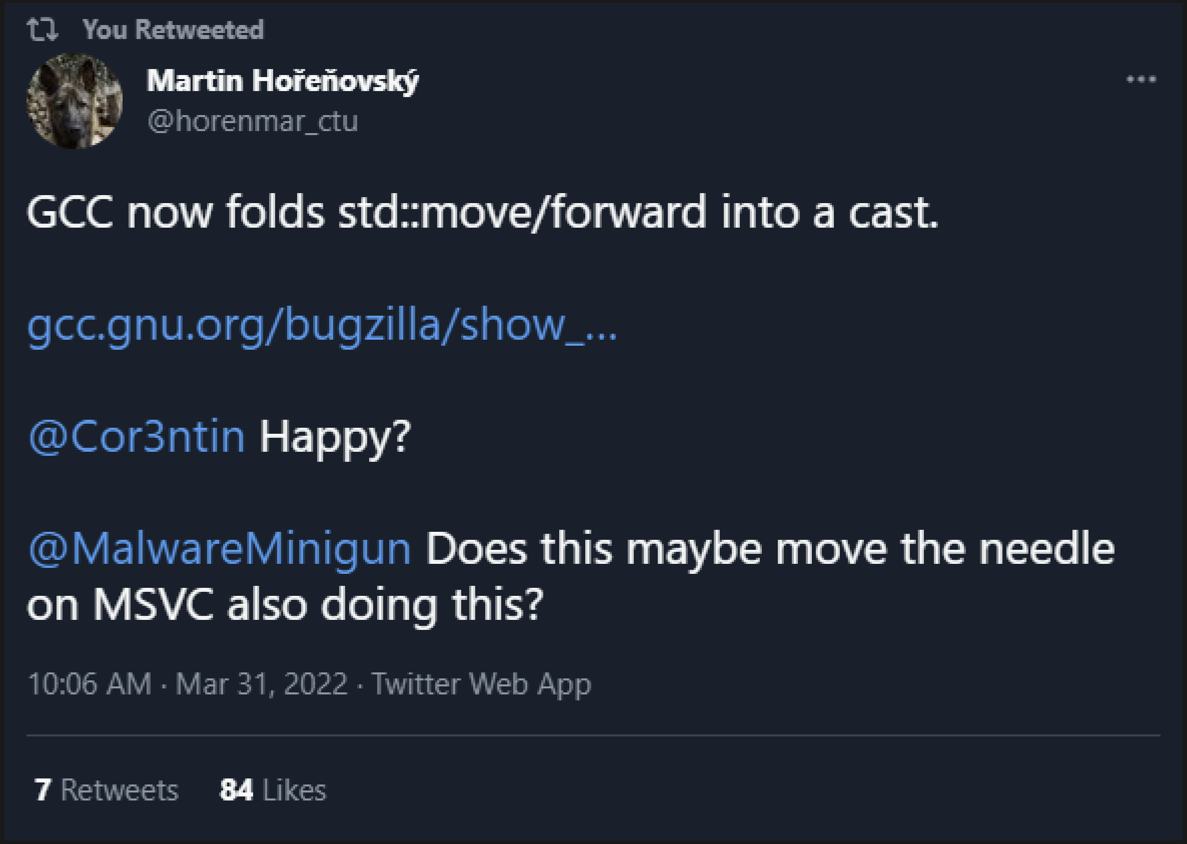
Some people resort to macros:

```
#define MOV(...) \
    static_cast< \
        std::remove_reference_t< \
            decltype(__VA_ARGS__)>&&>(__VA_ARGS__)

#define FWD(...) \
    static_cast<decltype(__VA_ARGS__)&&>(__VA_ARGS__)
```

(From https://www.foonathan.net/2020/09/move-forward/)

Some compilers are taking action:



You Retweeted

Martin Hořeňovský
@horenmar_ctu

GCC now folds std::move/forward into a cast.

gcc.gnu.org/bugzilla/show_...

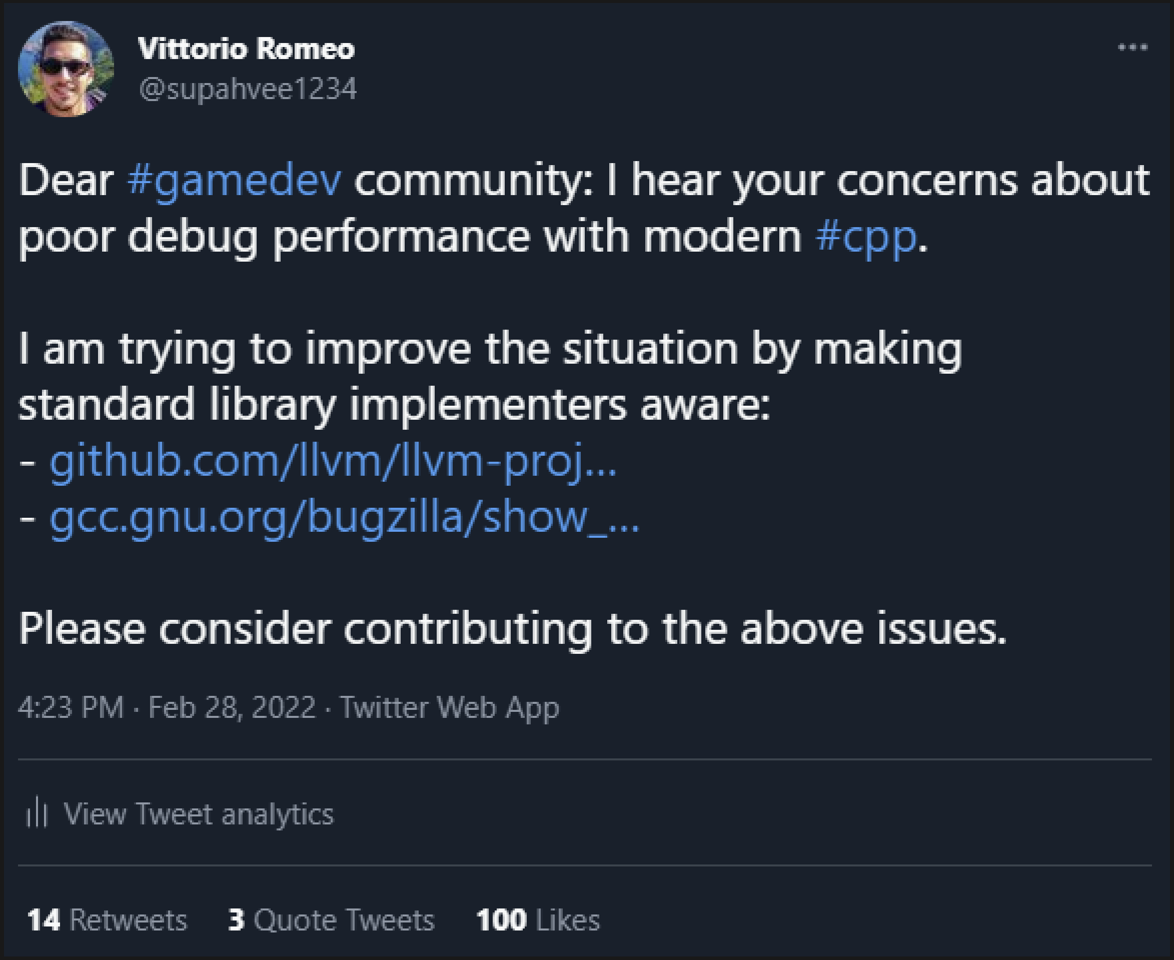
@Cor3ntin Happy?

@MalwareMinigun Does this maybe move the needle on MSVC also doing this?

10:06 AM · Mar 31, 2022 · Twitter Web App

7 Retweets **84** Likes

If this matters to you, make your voice be heard!



Vittorio Romeo
@supahvee1234

Dear #gamedev community: I hear your concerns about poor debug performance with modern #cpp.

I am trying to improve the situation by making standard library implementers aware:
- github.com/llvm/llvm-proj...
- gcc.gnu.org/bugzilla/show_...

Please consider contributing to the above issues.

4:23 PM · Feb 28, 2022 · Twitter Web App

14 Retweets **3** Quote Tweets **100** Likes

- Links:
  - GCC: Bugzilla Report
  - Clang: GitHub Issue
  - MSVC: Developer Community Feedback

Vittorio Romeo

mail@vittorioromeo.com

@supahvee1234

https://github.com/vittorioromeo/accu2022