

# C++ Club Meeting Notes

Gleb Dolgich

2019-06-27

# Concept-based interfaces, by Gianluca Delfino

[Article](#) | [Code](#) | [Godbolt](#) | [Reddit](#)

```
1 template <typename T>
2 concept Shape = requires(const T& t)
3 {
4     { t.area() } -> float;
5 };
6
7 template <typename T>
8 struct Rectangle
9 {
10     Rectangle() { static_assert(Shape<T>); }
11     float area() const;
12     T base;
13     T height;
14 };
```

## Concept-based interfaces, by Gianluca Delfino (cont.)

### Alternative solution:

```
1 template <typename T>
2 concept Shape = requires(const T& t)
3 {
4     { t.area() } -> float;
5 };
6
7 template<class T>
8 struct ModelsShape
9 {
10     ModelsShape() requires(Shape<T>) = default;
11 };
12
13 struct Circle: ModelsShape<Circle>
14 {
15     float area() const;
16     float radius;
17 };
```

- ▶ [FOSS Bytes](#)
- ▶ [Intel Announcement](#)
- ▶ [Reddit](#)

Part of Intel One API Project. Based on C++14 and SYCL. Open Source.  
Developer Beta in 2019 Q4.

## Follow-up: std::function const correctness

```
1 struct Callable {
2     void operator()(){count++;}
3     void operator()() const = delete;
4     int count = 0;
5 };
6
7 void f()
8 {
9     Callable counter;
10    std::function<void(void)> f = counter;
11    f();
12    const std::function<void(void) const> cf = counter;
13    //                               ^^
14    // error: implicit instantiation of undefined template
15    // 'std::__1::function<void () const>'
16    //
17    cf(); // Should not compile
18 }
```

## Follow-up: std::function movable callables

```
1 void f()
2 {
3     std::unique_ptr<int> up;
4     auto l=[up=std::move(up)](){};
5     std::function<void(void)> f1=l; // Error
6     std::function<void(void)> f2=std::move(l); // OK
7 }
```

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/#mailing2019-06>

[https://www.reddit.com/r/cpp/comments/c3mup9/c\\_precologne\\_mailing/](https://www.reddit.com/r/cpp/comments/c3mup9/c_precologne_mailing/)

## Direction for ISO C++ (R3)

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p0939r3.pdf>

https:

[//www.reddit.com/r/cpp/comments/c3mes0/direction\\_for\\_iso\\_c\\_r3/](https://www.reddit.com/r/cpp/comments/c3mes0/direction_for_iso_c_r3/)



# Proposal: Enumerating Core Undefined Behaviour (P1705R0)

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1705r0.html>

[https://www.reddit.com/r/cpp/comments/c4548m/a\\_proposal\\_to\\_enumerating\\_core\\_undefined\\_behavior/](https://www.reddit.com/r/cpp/comments/c4548m/a_proposal_to_enumerating_core_undefined_behavior/)

## bad\_alloc is not out-of-memory!

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1404r1.html>

TL;DR: Throwing `std::bad_alloc` is not the same as "there is no heap space available" - in particular when dealing with custom allocators.

## In support of P1485 “Better keywords for coroutines”

<https://quuxplusone.github.io/blog/2019/06/26/pro-p1485/>

<https://stackoverflow.com/a/44244451/1424877>

*A function becomes a coroutine by having [a keyword such as `co_await`, `co_yield`, or `co_return`] in its body. So [without close inspection of every line of the body] they are indistinguishable from functions.*

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1485r1.html>

[https://www.reddit.com/r/cpp/comments/c5uu56/in\\_support\\_of\\_p1485\\_better\\_keywords\\_for\\_coroutines/](https://www.reddit.com/r/cpp/comments/c5uu56/in_support_of_p1485_better_keywords_for_coroutines/)

# To boldly suggest an overall plan for C++23

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p0592r1.html>

Must have:


- ▶ Library support for coroutines
- ▶ Executors
- ▶ Networking


Good to have:


- ▶ Reflection
- ▶ Pattern matching


- ▶ P1662R0 Adding async RAI support to coroutines
- ▶ P1678R0 Callbacks and Composition
  - ▶ <https://github.com/ReactiveX/RxCpp>
  - ▶ <https://github.com/facebookresearch/pushmi>
- ▶ P1688R0 Towards a C++ Ecosystem Technical Report
- ▶ P1711R0 What to do about contracts?
- ▶ P1717R0 Compile-time Metaprogramming in C++
- ▶ P1729R0 Text Parsing
  - ▶ <https://github.com/eliaskosunen/scnlib> (Apache-2.0) | [Reddit](#)


# Twitter


**Stuart Dootson** @studoot 7 hrs  
@jckarter Is a `dynamic_cast` an `ex_static_cast`?

**psu\_13** @psu\_13 13 hrs  
@jckarter `sarcastic_cast<T>(x)`

**Amro Mousa** @amdev 13 hrs  
@jckarter Your Tweets are so generic

**Jasdev Singh** @jasdev 14 hrs  
@jckarter "Live fast, `unsafeBitCast(_:to:)'`" - Ancient Proverb  
From Newark, NJ`

**Steve Canon** @stephentyrone 14 hrs  
@jckarter `// Discards x if it does not spark joy.`

**Joe Groff** 5 Replies  
@jckarter

`ecstatic_cast<T>(x)`  
19/02/2019, 23:55 (Yesterday)  
Twitter Web Client

31 Likes2 RetweetsThread >