C++ Club Meeting Notes

Gleb Dolgich

2018-03-01

C++ Developer survey

Link

Prof. Liskov receives 2018 Computer Pioneer Award

Web

Prof. Barbara Liskov Selected to Receive the 2018 IEEE Computer Society's Computer Pioneer Award. The award is given for significant contributions to early concepts and developments in the electronic computer field, which have clearly advanced the state-of-the-art in computing. Liskov is being recognized "for pioneering data abstraction, polymorphism, and support for fault tolerance and distributed computing in the programming languages CLU and Argus."

P0581R1: Standard Library Modules

P0581R1

- Leave “C headers” alone
- Present modules as logical sets of functionalities
- Robust support for C++ in diverse environments
- Every standard facility is provided by exactly one module

As a first approximation, we suggest the following:

- `std.fundamental`
- `std.core`
- `std.math`
- `std.io`
- `std.concurrency`
- `std.os`
- `std`

P0942R0: Introducing a <smart_ptr> header

P0942R0

P0122R6: span: bounds-safe views for sequences of objects

- P0122R6
- P0546r2 : Span - foundation for the future

The span type is an abstraction that provides a view over a contiguous sequence of objects, the storage of which is owned by some other object.

Requires C++11. A reference implementation is available as part of Microsoft GSL.

P0201R3: `polymorphic_value`: a polymorphic value-type for C++

P0201R3

Add a class template, `polymorphic_value<T>`, to the standard library to support polymorphic objects with value-like semantics. Copies are deep and `const` is propagated to the owned object.

See also:

`value_ptr`—The Missing C++ Smart-pointer

p0907r0: Signed Integers are Two's Complement

- p0907r0
- Reddit

P0645R1: Text Formatting

P0645R1

This paper proposes a new text formatting library that can be used as a safe and extensible alternative to the `printf` family of functions. It is intended to complement the existing C++ I/O streams library and reuse some of its infrastructure such as overloaded insertion operators for user-defined types.

Example:

```
1 string message = fmt::format("The answer is {}.", 42);
```

A full implementation of this proposal is available at https://github.com/fmtlib/fmt.

P0671R1: Parametric functions

> *Please don't assume that you know what this paper is about. <...>*
> *I am almost certain: you haven't seen this before.*

P0671R1: Allow naming arguments in regular index-based, C++-style function calls, but ignoring them from the standard's point of view (but not necessarily so from the implementers')

```
1 double Gauss(double x, double mean = 0.,
2              double width = 1., double height = 1.);
3 Gauss(0.1, mean := 0., width := 2.); // same as below
4 Gauss(0.1, 0., 2.);
```

p0745r0: Concepts in-place syntax, by Herb Sutter

p0745r0

Concepts TS and P0694R0:

```
1 void Sort(Sortable& s);
```

Proposed:

```
1 void Sort(Sortable{}& s);
```

p0873r1: A plea for a consistent, terse and intuitive declaration syntax

p0873r1

- Allow `auto` as a parameter in functions
- Consistent use of `auto` and `typename`
- Allow multiple constraints for each type
- Allow use of concepts consistently and uniformly

P0915R0: Concept-constrained auto, by Vittorio Romeo and John Lakos

P0915R0

```
1 template <typename T>
2 void foo(std::vector<T>& v)
3 {
4     auto<RandomAccessIterator> it{std::begin(v)};
5     // ...
6 }
```

p0956r0: Answers to concept syntax suggestions, by Bjarne Stroustrup

p0956r0

Batteries not included: what should go in the C++ standard library? (by Guy Davidson)

Post

P0267R7: A Proposal to Add 2D Graphics Rendering and Display to C++

"A graphics library shouldn't be part of the standard library. The standard library is too big already, and this will make it so much bigger. It's just not appropriate. The library should only contain small types."

A cake for your cherry: what should go in the C++ standard library? (by Corentin)

- Post
- Reddit

I do think this is a path that should not be pursued and that doing so would be, at best, a waste of time.

The question then becomes : can the committee spare the time to work on a 2D graphics library and is that a priority ?

In its current form, which is limited to drawing shapes, the proposal is about 150 pages long. It's one of the biggest proposal submitted for the next meeting.

What Should Go Into the C++ Standard Library (by Titus Winters)

Post, Reddit

C++ cannot afford to be all things to all people <...> why are we thinking that a Graphics library to make it easy to visualize things for novices is a good fit?

So, what should go in the standard library? Fundamentals.

Graphics won't make C++ a teaching language - learning to be an algorithmic thinker should be done in a language that has fewer sharp edges. C++ won't make Graphics easy, and tying any sort of Graphics API to the legacy constraints of C++ won't make for a great API in that space.

There needs to be some mechanism to readily distribute libraries and dependencies in the C++ community.

Blaze C++14 maths library

- BitBucket (BSD)

Blaze is an open-source, high-performance C++ math library for dense and sparse arithmetic. With its state-of-the-art Smart Expression Template implementation Blaze combines the elegance and ease of use of a domain-specific language with HPC-grade performance, making it one of the most intuitive and fastest C++ math libraries available.

Videos by Qt Studios

YouTube

C++ Dublin Users Group

Videos on YouTube

DHH on Danes

DHH @dhh

We are born with a native tongue hardly anyone speaks, and that is needlessly hard to learn. Inventing programming languages and frameworks is the only way to get the world to think in our words 😄

> 
> David R. P. Morris @davidrpmorris
>
> Has anyone noticed how many popular programming languages and frameworks were developed by Danes? C++ (@stroustrup), PHP (@rasmus), Rails (@dhh), C# (@ahejlsberg)
>
> 22h • 14/02/2018 • 13:56

21h • 14/02/2018 • 14:30

Serge Kruppa @sergejf

@dhh

Another example: Borland, of Turbo Pascal fame, was founded in 1981 by three Danish citizens: Niels Jensen, Ole Henriksen, and Mogens Glad. Not strictly new prog languages, but a huge impact on software dev at the time!

20h • 14/02/2018 • 16:08