C++ Club Meeting 100

Gleb Dolgich

2020-02-27

Prague Trip Reports

- Bryce Lelbach et al.
  - 252 attendees! 23 subgroups! 9 tracks!
  - **C++20** is done!
  - **C++23** roadmap! **Standard library modules**; **library support for coroutines**; **executors**; **networking**. Also: reflection; pattern matching; contracts.
- Herb Sutter
  - Reddit
- CppCast with Hana Dusíková
  - YouTube

C++20 is here

- Video
- Reddit

Bjarne Stroustrup on C++20’s significance

- ISO C++
    - Reddit

Quote:

- 30 years of C++ standardization.
- 40 years of C++.
- C++20 is the 6th standard, the 3rd major standard; by “major” I mean “changes the way people think.”
- This is something like the 75th meeting; I have been at about 70 of those.

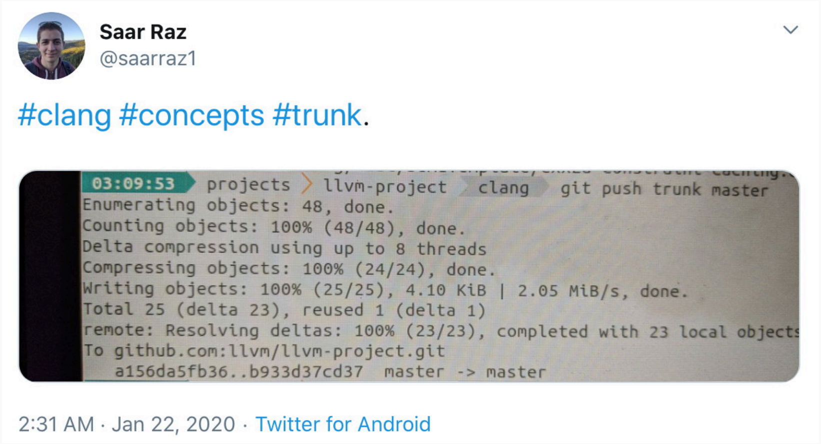
Excited about C++20

- Reddit

Comment:

> *I work in a large codebase that was originally written in C and was compiled with C++03 just a few years ago. Since then, we have upgraded through C++11, C++14 and are now using C++17. So far, my experience is that every upgrade has been almost exclusively a positive experience, and each version has made it easier to write safe and expressive code.*

Concepts pushed to Clang master



Saar Raz
@saarraz1

#clang #concepts #trunk.

```
03:09:53  projects > llvm-project  clang  git push trunk master
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (25/25), 4.10 KiB | 2.05 MiB/s, done.
Total 25 (delta 23), reused 1 (delta 1)
remote: Resolving deltas: 100% (23/23), completed with 23 local objects
To github.com:llvm/llvm-project.git
   a156da5fb36..b933d37cd37  master -> master
```

2:31 AM · Jan 22, 2020 · Twitter for Android

- Reddit

[C++ coroutines] Initial implementation pushed to GCC master

- Message
- Reddit

> *This is not enabled by default (even for* ***-std=c++2a****), it needs* ***-fcoroutines***

How to keep up with C++ news

- Reddit
    - blog posts
    - code reviews
    - follow the C++ tag on StackOverflow
    - follow C++ conference talks
    - cppreference.com
    - books
    - read proposals
    - join the commitee!
    - don’t…

CppCast Ep. 233: Large Scale C++ with John Lakos

- Audio
- Video

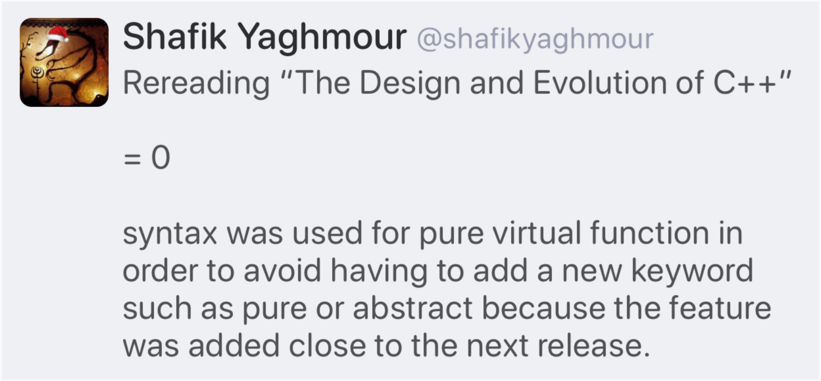
> *There's a misunderstanding (of contracts in the C++ committee – GD) that's not easy to appreciate if you're not a real day-to-day software engineer. That is what derailed contracts. I will fix it. I promise you, I will fix it.*

Follow-up: Aggregates

From CppReference: An aggregate is one of the following types:

- array type
- class type (typically, `struct` or `union`), that has
  - no private or protected *direct* (since C++17) non-static data members
  - no *user-declared constructors* (until C++11)
  - no *user-provided constructors* (explicitly defaulted or deleted constructors are allowed) (since C++11) (until C++17)
  - no *user-provided, inherited, or explicit constructors* (explicitly defaulted or deleted constructors are allowed) (since C++17) (until C++20)
  - no *user-declared or inherited constructors* (since C++20)
  - no virtual, private, or *protected* (since C++17) base classes
  - no virtual member functions
  - no *default member initializers* (since C++11) (until C++14)

Twitter: Pure virtual function syntax (1/2)



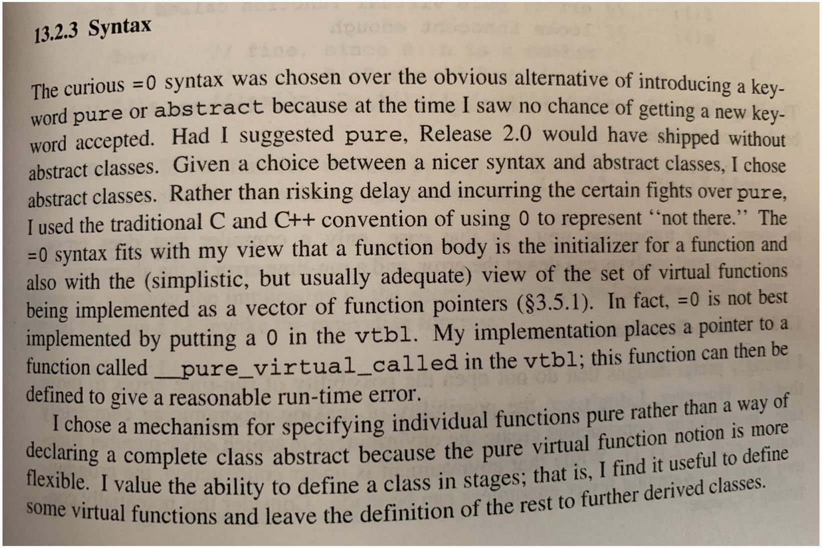
Shafik Yaghmour @shafikyaghmour

Rereading "The Design and Evolution of C++"

= 0

syntax was used for pure virtual function in order to avoid having to add a new keyword such as pure or abstract because the feature was added close to the next release.

Twitter: Pure virtual function syntax (2/2)



13.2.3 Syntax

The curious =0 syntax was chosen over the obvious alternative of introducing a keyword `pure` or `abstract` because at the time I saw no chance of getting a new keyword accepted. Had I suggested `pure`, Release 2.0 would have shipped without abstract classes. Given a choice between a nicer syntax and abstract classes, I chose abstract classes. Rather than risking delay and incurring the certain fights over `pure`, I used the traditional C and C++ convention of using 0 to represent ‘‘not there.’’ The =0 syntax fits with my view that a function body is the initializer for a function and also with the (simplistic, but usually adequate) view of the set of virtual functions being implemented as a vector of function pointers (§3.5.1). In fact, =0 is not best implemented by putting a 0 in the `vtbl`. My implementation places a pointer to a function called `__pure_virtual_called` in the `vtbl`; this function can then be defined to give a reasonable run-time error.

I chose a mechanism for specifying individual functions pure rather than a way of declaring a complete class abstract because the pure virtual function notion is more flexible. I value the ability to define a class in stages; that is, I find it useful to define some virtual functions and leave the definition of the rest to further derived classes.

Move, simply

- Herb Sutter
    - HackerNews

> *The state of `a` after it has been moved from is the same as the state of `a` after any other non-const operation. Move is just another non-constfunction that might (or might not) change the value of the source object.*

- *Move, even more simply* by Corentin Jabot

> *In the absence of other information, do not do anything to an object on which `std::move` has been called, except assignment operator and destructor.*

Rust is better than C++20, by David Sankel

David Sankel, “We Have C++20” bloopers:

> *(C++) is like Rust, but worse.*

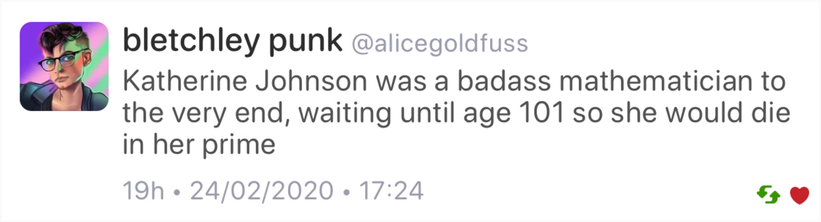
Operator:

> *What’s better about Rust?*

David Sankel:

> *I don’t know, I haven’t actually used Rust.*

Twitter: Katherine Johnson



bletchley punk @alicegoldfuss

Katherine Johnson was a badass mathematician to the very end, waiting until age 101 so she would die in her prime

19h • 24/02/2020 • 17:24